AD-776 236

INTERACTIVE SYSTEMS RESEARCH

Morton I. Bernstein

System Development Corporation

Prepared for:

Advanced Research Projects Agency

30 November 1973

UNCLASSIFIED
Security Classification

AD 776236

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) System Development Corporation 2500 Colorado Avenue Santa Monica, California 90406	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE
Interactive Systems Research: Final Report to the Director, Advanced Research Projects Agency, for the Period 16 September 1972 to 30 September 1973

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical

5. AUTHOR(S) (First name, middle initial, last name)
Morton I. Bernstein

6. REPORT DATE 30 November 1973	7a. TOTAL NO. OF PAGES iii, 71	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. DAHC15-73-C-0080 b. PROJECT NO. 3D30 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) TM-5243/000/00 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY ARPA/Information Processing Techniques

13. ABSTRACT

This report documents research in five areas of applied computing: (1) speech understanding, (2) the use of inference in data management, (3) image and signal processing (4) network research and development (including ARPA Network activities), and (5) network data management. Research activities reported upon include the development and demonstration of a prototype Voice Data Management System; the design of an interactive, graphics-oriented system for use by image analysts; and several approaches to amplifying the power and flexibility of the data management process and data management languages. All of these activities are focused on the man-machine interface and specifically on ways of making it more natural and adaptable to the needs and preferences of users.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

TM-5243/000/00

INTERACTIVE SYSTEMS RESEARCH: FINAL REPORT TO THE DIRECTOR, ADVANCED RESEARCH PROJECTS AGENCY, FOR THE PERIOD 16 SEPTEMBER 1972 TO 30 SEPTEMBER 1973

30 NOVEMBER 1973

M. I. BERNSTEIN
(213)393-9411, EXT 6117

THIS REPORT WAS PRODUCED BY SDC IN PERFORMANCE OF CONTRACT DAHC15-73-C-0080, ARPA ORDER NO. 2254, PROGRAM CODE NO. 3D30.

ia

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND SUMMARY

This document constitutes the final report to the Advanced Research Projects Agency (ARPA) for the period 16 September 1972 through 30 September 1973 on System Development Corporation's (SDC's) research and development program in Interactive Systems Research (ISR). The ISR program is, in the main, a continuation of selected projects from the previous ARPA-sponsored Computer-Assisted Planning (CAP) program, the emphasis having changed from the eventual development of an interactive planning system to concentration on the development of more basic technology for improved man-machine interactive systems with applications to a variety of anticipated military needs.

The ISR program, with its emphasis on improving the man-machine interface, contains the potential for both enhancing the utility and improving the productivity of total systems (encompassing the hardware, the software, and the human beings) and bringing better service to the non-professional user of computer-based services. It is our hypothesis that putting the ultimate user of information (and the processes by which it is transformed to be more meaningful to him) in direct contact with the information base (and the transformation process) in a mode that is normal and natural to his training and discipline can enhance productivity and creativity at all operational levels. To validate this hypothesis, prototype systems for communicating with information processing systems in ordinary English, by voice, and by means of hand-drawn images and hand-printed characters are in various stages of development.

To complement the activities that directly improve the man-machine interface and to round out the program, projects in network research and development and network data management are also being carried out.

There are significant deviations between the program that was originally proposed and accepted by ARPA and the one that was actually performed; this resulted from the discussions and recommendations by the Contract Technical Monitor at mid-year. Therefore, although the continuity from CAP to ISR was originally to have been transitional, it was somewhat more abrupt for some projects.

The following summarizes these activities for the past year.

1.1 SPEECH UNDERSTANDING RESEARCH

The continuing long-term goal of SDC's Speech Understanding Research project is to develop and implement a data management system that is controlled and operated by its users through free-form spoken English. The project has two major research focuses: (1) the processing of acoustical data contained in the speechwave and (2) the application of linguistic procedures and contextual information to make the processed speechwave intelligible. Because these two focuses are of virtually equal importance, the project during the year devised a means of proceeding on both in parallel by implementing two versions of the prototype speech understanding system. One version incorporated a well-developed acoustic processor; the other incorporated a well-developed linguistic component. This permitted a comprehensive group of experiments to be undertaken in a successful effort to validate the feasibility of the project's original technical approach to the speech understanding problem.

1.2 DEDUCTIVE INFERENCE RESEARCH

A serious limitation of English question-answering systems has been their inability to draw inferences from (a) a user's query and (b) the content of a data base in order to derive meaningful answers that are not stored in the data base in explicit form. This limitation has kept the promise of general-purpose use of English-based systems from being realized. A key technical problem, which has not been solved in any previous approach to mechanizing inference, is that of selecting from among a large file of facts and general assertions the very few that are relevant for a particular deductive operation. The design presently being implemented by SDC incorporates interacting files and processors that are specifically designed to speed up and simplify this selection process. The design also incorporates an advanced Intermediate Language (IL2) that serves the dual role of being both a target language enabling precise representation of the meaning of English statements and a data management and deduction-driver language capable of controlling the storage and processing of relational data and general data from which inferences are to be drawn.

During the year, two major processors were programmed in LISP, and considerable theoretical work was done to establish the logical bases for efficiency in deductive processing of large data files. The year's work was not closely tied to general advancements in the CONVERSE system, as had originally been proposed, owing to the technical redirection mentioned above. This amounted to a reduction in scope, but an increase in the intensity with which the approach was developed and tested.

1.3 IMAGE AND SIGNAL PROCESSING

After conducting research in basic interactive computational graphics technology for several years--research primarily into techniques for efficient automatic recognition and evaluation of hand-printed mathematical expressions--SDC during this past year studied the feasibility of applying this technology to a practical problem area, that of interpreting optical or electronic images produced in the course of scientific experiments, medical tests, surveillance operations, and satellite and telescopic photography. The feasibility study was undertaken because the earlier research had demonstrated that flexible man-machine interaction via hand-printed graphics was possible and could be adapted to the amelioration of a particularly costly, slow, and cumbersome--but important and necessary--application. Image processing is such an application.

During the year, the Image and Signal Processing project studied the state of the art, formulated a comprehensive set of requirements for a facility to support interactive image processing research and analysis, and drew up design and language specifications for a system that will meet those requirements. These activities were extensively documented in a series of formal reports and working papers.

1.4 NETWORK RESEARCH AND DEVELOPMENT

Major changes were made in SDC's computer laboratory and operating-system facilities during the year. SDC is now running, and will soon have available to the ARPANET, the IBM Virtual Machine (VM) system operating on a 370/145 computer. The transition to VM-370 from the earlier ICOS operating system necessitated changes in the ARPANET connection configuration; after a survey of Network Control Program (NCP) installations around the ARPANET, SDC selected UC Santa Barbara's NCP as the most feasible for reimplementation on the 370/145 under VM-370. Modifications were made to the UCSB NCP to make it compatible with VM-370, and a virtual machine will support the NCP. This will offer all the facilities of a virtual 370 to an ARPANET user.

1.5 NETWORK DATA MANAGEMENT

The three projects in the network data management area--English Data Access, Network Data Sharing, and Common Information Structures--are closely related by their common emphasis on techniques for more flexible access to computerized data bases. The English Data Access project and the Network Data Sharing projects evolved out of work in earlier years on the CONVERSE English data management system. Both projects, however, have diverged considerably from the original direction of the CONVERSE work. The English Data Access project was created at the time of the mid-year contract technical redirection in an effort to adapt the insights and experience gained by the CONVERSE researchers to the immediate problem of natural-language access existing data management systems. An approach to this problem, called conceptual processing, was developed. A Conceptual Processor was described that would function as an adjunct to a specific data management system, mediating between the system and an English-speaking user. This processor would accept free-form English queries, but only if they were comprehensible--that is, conceptually meaningful or verifiable--in terms of the semantic content of the target data base and only if they express relations that match functions that can be performed by the target data management system.

The Network Data Sharing project addressed a different problem: that of providing a community of computer-network users with mutual access to data residing in the variety of differently structured data management systems found in a non-uniform network such as the ARPANET. There were two problems to be dealt with: (1) translating from one data management query language to another and (2) transforming data from one structure (e.g., hierarchical) to another (e.g., relational). There were also several possible approaches to dealing with these problems, and during the year preceding this one the project selected what it called the Integrated approach. The Integrated approach involves a common data-sharing language and, associated with each data management system on the network, a translation interface that would translate a query in the common language into one or more queries in the language of the target system. The common language that was formulated, called Intermediate Language for Data Sharing (ILDS), incorporates features that accommodate the likely case of translations involving incompatible data structures. In addition to the common language, the project specified the tasks of the translator interfaces.

The English Data Access and Network Data Sharing projects both concluded this year. The third in this group, Common Information Structures, will continue into the next contract year. It is directed at the problem of transforming or shifting data bases from one data management system to another; this contrasts with the Network Data Sharing project's objective of accessing a remote data base via a common language. The project is focusing on methods of describing data structures and data bases in a logical form, so that the logical description might be used as a vehicle for transforming data into a particular physical-orgnizational form to suit a particular data management system's storage characteristics. During the year, a formal schema was developed for transforming a data base from a system S_i to a system S_j through a logical file-description mechanism. The major emphasis during the coming year will be on characterizing and implementing data base transformers.

2. SPEECH UNDERSTANDING RESEARCH

The continuing long-term goal of SDC's Speech Understanding Research project is to develop and implement a data management system that is controlled and operated by its users through free-form spoken English. The short-term goal is the construction and refinement of a limited system that can accept continuous speech and is demonstrably usable by at least two speakers, but in which limitations are imposed on both the vocabulary and syntax of the English subset permitted. Our basic approach to automatic speech understanding is embodied in a concept we call Predictive Linguistic Constraints (PLC). The resulting system will be a Predictive Linquistic Constraints-Vocal Data Management System (PLC-VDMS) [1].

PLC-VDMS is a complex set of processing modules organized into two principal components--a linguistic "top end" and an acoustic "bottom end." The top end embodies the PLC concept. It contains the data management system and the modules that use syntax, semantics, thematics, and other nonacoustic considerations to generate predictions of words that may be contained in the input utterance being processed. The PLC concept is based on the assumption that it is easier to verify or reject predictive hypotheses about the nature of a particular set of data than it is to generate, in the absence of any other information, the consequence of those data. Predictive hypotheses are generated from an analysis of the user's goal-directed behavior and from thematic considerations. The system then searches for recognizable fragments of an input utterance and, from them, generates predictive hypotheses about the contents or meaning of other parts of the utterance. PLC-VDMS differs markedly from most other transducers or compiler-like systems in that fixed directionality is replaced by a heuristic assembly of top-down, bottom-up, right-to-left, and left-to-right scanning strategies operating in parallel on the same or different segments of the input utterance. Two major advantages are obtained from this: (1) the system may start working on the least ambiguous portions of the input and (2) predictions need not be limited to near neighbors of a recognized input segment but may be applied to any portion of the entire utterance.

The bottom end contains the processes that extract information from the speech signal and make acoustic-phonetic labeling decisions. A human phoneticist's transcription of an utterance contains elements of subjectivity reflecting the transcriber's personal experience with the language and the judgments required to map an extremely large variety of speech sounds into a relatively small set of transcription symbols. The acoustic bottom end we are developing reflects this premise in that each acoustic segment is multiply labeled, and each label is assigned a probability.

2.1 PROGRESS

In order to take maximum advantage of time and project resources, work on the top and bottom ends of PLC-VDMS has proceeded in parallel, and two separate versions of the system have been constructed and tested: one with a well-developed PLC top end (Version A), and the other with a well-developed acoustic bottom end (Version B). Version A operates on the Voice I/O Laboratory Raytheon 704 minicomputer in conjunction with the IBM 370/145 and incorporates the system architecture envisioned for the first large scale PLC-VDMS system. Version B operates entirely on the Raytheon 704 and for that reason is referred to herein as the "mini" system; it has served as a vehicle for acoustic-phonetic research. In assembling the first full-scale version of PLC-VDMS, we will combine the best elements of both of these initial versions.

Both versions of PLC-VDMS have been tested with utterances that conform to the permitted vocabulary and syntax of the system and the contents of a data base of information on U.S., U.S.S.R., and U.K. submarine fleets. These versions have achieved acceptable results on a reasonably large set of utterances.

2.1.1 PLC-VDMS System Architecture

2.1.1.1 CSL and MCP

The systems architecture of Version A of PLC-VDMS is unique. A control structure language (CSL) has been developed that describes all control and data flow paths among the major components in the system. The CSL compiler and run-time package have been implemented in the SDC LISP 1.5 on SDC's IBM 370/145 computer. Compilation of the control structure description produces a main control program (MCP) that invokes modules at appropriate times and handles all inter-data communications.

CSL allows both declarative and imperative statements. The declarative portions of the language allow definitions of modules, busses, and global stores. The busses are the PLC software analog to the equivalent hardware concept. A bus provides a dynamic short-term data path among all modules connected to it. Data that need to be available for extended periods or time are located in global stores. The busses and global stores are accessed functionally with routines provided by the MCP. Since CSL provides for simultaneous execution of modules, data updates are synchronized by the MCP. A module is a program that has no arguments, returns no values, and may not directly invoke any other module. All data communication between a module and its environment is through the MCP. Further, module instantiation is done only by the MCP. The declarations of a module include the names of the busses it reads and writes, the names of the global stores it interrogates and updates, and alias names by which the module may be known.

The imperative statements in CSL allow specifications of order dependencies of execution among the modules. Synchronization points for updating the busses and common stores are included. Furthermore, the user may include direct code in the CSL description in addition to module-invocation directives. Debugging and trace statements may be added or deleted from a CSL program without modifying the main body of the program.

CSL has proved a valuable tool for testing and debugging the PLC top end. For example, test cases have been run through the system for performance comparisons with various modules "unplugged." This testing was accomplished simply by modifying CSL statements, not the modules. The major virtue of CSL, however, is the availability in one place of a concise statement of all control and data dependencies among major parts of a system.

2.1.1.2 DMS

The data management module, DMS, handles structured queries on a relational data base. It receives input and produces output in orthographic representation. The DMS language has four major classifications of input statements. The classifications, members of each class, and examples of DMS statements are shown in Figure 2-1. (The language has the ability to describe a report but no facilities to produce one, since this has no bearing on speech research.) DMS generates the correct response to all legal questions, and this information is used by other modules to create PLC's view of the user's interest "set."

2.1.1.3 PLC Top End

The PLC top end comprises two major groups of modules, GLOBAL and LOCAL. There are four LOCAL modules:

SYNT--A top-down parsing program that generates partial hypotheses on what to look for next.

SYNB--A bottom-up parsing program that applies non-syntactic constraints to partial constructions.

SYNS--A decision-making program that decides whether to go bottom up or top down to continue processing of a partial parse.

CLSF--Interfaces the output of acoustic-processing (ACU) to the local modules.

The overall PLC-VDMS configuration containing these four modules is shown in Figure 2-2.

SYSTEM CONTROL

LOGIN	LOGIN ONE THREE TWO
LOGOUT	LOGOUT
ENTER	ENTER QUERY MODE

GENERAL INFORMATION

EXPLAIN	EXPLAIN LOGIN
DESCRIBE	DESCRIBE
SHOW	SHOW COUNTRY

INTERACTIVE QUERY MODE

PRINT	PRINT LENGTH BEAM AND DRAFT WHERE COUNTRY EQUALS U.S.A.
COUNT	COUNT WHERE TORPEDO TUBES GREATER THAN ONE ZERO
REPEAT	REPEAT WHERE COUNTRY EQUALS U.S.S.R. OR CLASS EQUALS ATTACK
TALLY	TOTAL MISSILES WHERE SAME
SUBSET	SUBSET WHERE SUBMERGED DISPLACEMENT LESS THAN SEVEN FIVE

REPORT DESCRIPTION MODE

TITLE	TITLE IS MISSILES REPORT
HEADINC	HEADINGS ARE TORPEDO TUBES MISSILES AND COUNTRY
CONTENT	CONTENTS ARE SAME
QUALIFY	QUALIFY WHERE MISSILES GREATER THAN OR EQUALS THREE
SORT	SORT BY MISSILES
TOTAL	TOTAL MISSILES
REVIEW	REVIEW
DELETE	DELETE TITLE

Figure 2-1. DMS Language

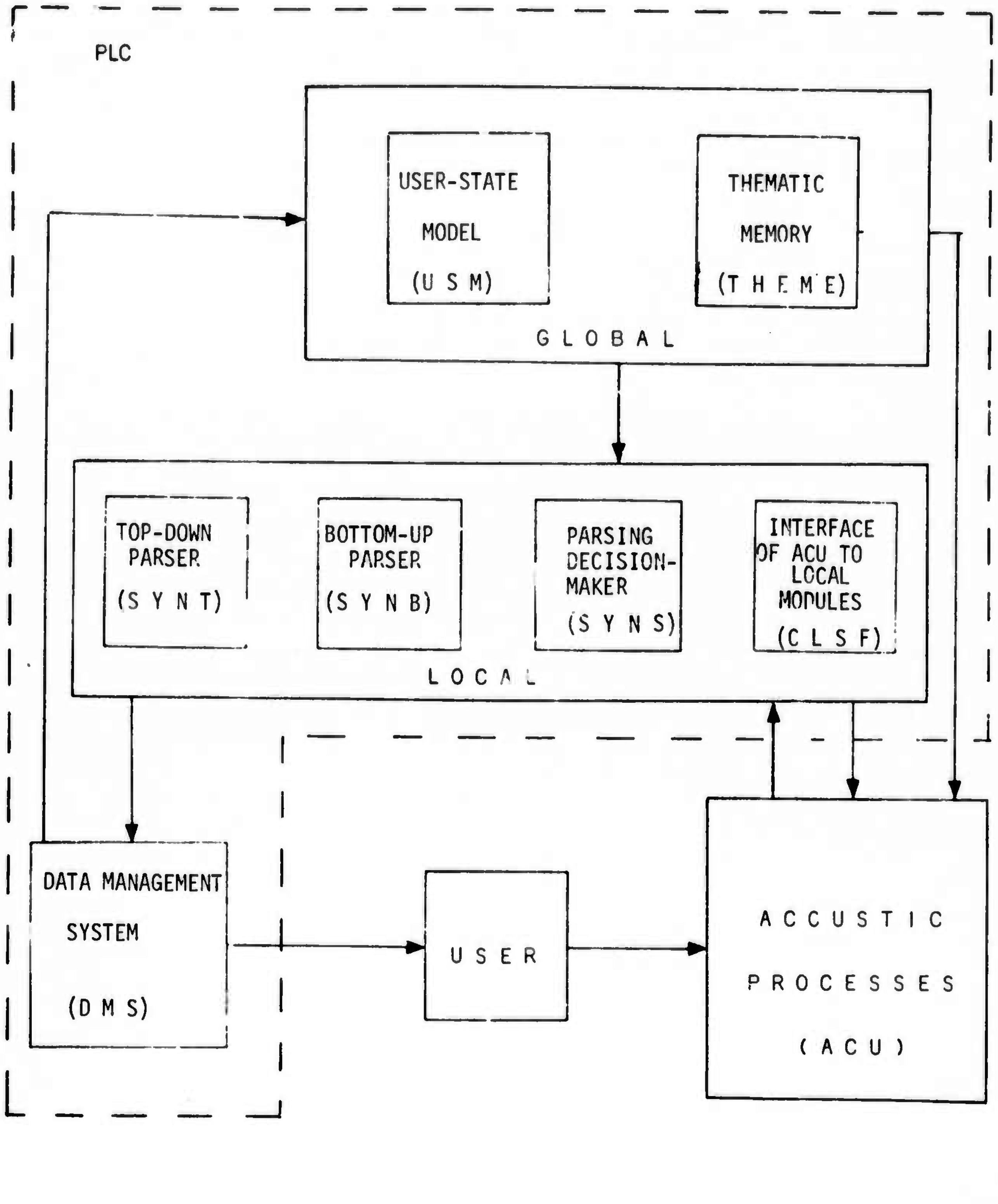

PLC
USER-STATE MODEL (U S M)
THEMATIC MEMORY (T H E M E)
GLOBAL
TOP-DOWN PARSER (S Y N T)
BOTTOM-UP PARSER (S Y N B)
PARSING DECISION-MAKER (S Y N S)
INTERFACE OF ACU TO LOCAL MODULES (C L S F)
LOCAL
DATA MANAGEMENT SYSTEM (D M S)
USER
ACCUSTIC PROCESSES (A C U)

Figure 2-2. PLC-VDMS Configuration (Version A)

In operation, LOCAL modules may be handled two or more trial parsings in parallel. This "breadth-first" approach to processing speech is advantageous for several reasons. First, it eliminates the need for large amounts of backup. Second, to priority-rank a "depth-first" search is computationally difficult within the present state of the art in acoustic-phonetic processing. In addition to the parallel, "breadth-first" processing, the LOCAL modules can go left to right, right to left, or skip over parts of the utterance to continue the recognition process.

The GLOBAL modules of PLC comprise the thematic memory (THEME) module and the user-state model (USM). THEME tracks the user-computer dialogs and attempts to predict words that are highly likely to occur in input utterance *n+1*, given that utterance *n* has been processed. Factors examined from the utterance *n* are such things as its content words, the non-numeric system responses to it, frequency of use or occurrences of content words, and the elapsed time since occurrence of such words. As words are introduced to THEME, they are combined with those already present. Redundant entries are noted and combined. As time passes, entries are aged and are eventually dropped from consideration.

USM is essentially a one-stage Markovian model for predicting the grammatical forms most likely to be used in the next input utterance. There are two important exceptions to the one-stage Markovian model: (1) in report-generation mode, all previous report-description commands are used in generating the syntax predictions; (2) after general information input, or after errors of the class "I DON'T UNDERSTAND YOU," the process becomes a two-or-more-stage Markovian model.

The following experiment illustrates the flexibility of PLC-VDMS Version A as a research vehicle. A 50-utterance input set was passed through the system under four different conditions. First, USM and THEME were both active; next, only USM was active; next, only THEME was active; finally, both were turned off. The results are summarized in Table 2-1, where "wrong" meant an incorrect recognition and "missed" meant that the utterance was not understood at all.

TABLE 2-1. PERFORMANCE OF PLC IN FOUR EXPERIMENTS FOR A 50-UTTERANCE INPUT SET

	USING THEME AND USM	USING USM	USING THEME	USING NEITHER
CORRECT	64%	62%	60%	54%
WRONG	10%	8%	18%	18%
MISSED	26%	30%	22%	28%
	100%	100%	100%	100%

2.1.1.4 The Bottom End

The acousti. bottom end constructed for PLC-VDMS Version A has these key features:

- Parameter weighting and inspection algorithms are entirely and automatically generated by the computer.
- The system incorporates a large, though not comprehensive, set of coarticulation rules in the dynamic mapping algorithms.
- Most of the bottom end runs in fail-soft mode.
- Some crude prosodies are already incorporated.
- The system design allows for easy incorporation of new rules and parameters as they are developed.

The Version A acoustic component has two modules, CWIP and CWIX. CWIP trys to locate a predicted word within given time boundaries. CWIX is similar to CWIP except that it works with a mutually exclusive set of words--for example, the digits. In both cases, the prediction includes a confidence limit that guides the relaxing or tightening of various constraints.

Two routines are associated with each phoneme occurring in the DMS language. One routine is used for establishing the existence of the phoneme given a right time boundary, the other given a left time boundary. For the vowels, diphthongs, and the three fricatives S, Z, and SH, a third routine may be used to establish the existence of the phone given no rigid time boundaries except at the word or phrase level. These three routines are used to establish anchor points when the parsing algorithms decide to skip over an area or when THEME requests a search for a particular word anywhere in the utterance. The symbols WL, WR, and WB are used in addition to the ARPABET;[1] they stand for left word boundary, right word boundary, and lexical interior word boundary, respectively. As an example, our phonetic spelling of "SORT BY" is "WL S OW R T WB B AY WR."

Each phoneme-associated routine contains a set of coarticulation rules depending upon the environment in which the phoneme is used. Figure 2-3 shows examples of the dynamic boundary-finding procedure. In the first example, a part of the word "modified," spelled M AA DX IX F AY D, is shown. As an illustration, the system attempts to locate boundaries of DX after AA. Failing, it then successfully establishes a boundary for DX within the boundaries of AA. The system notes that the IX to the right of DX is a reduced vowel, and an attempt is made to locate IX near the left boundary of DX. The second example, TYPE, shows that the system is looking for S-like segments representing burst release to the right of T. Transitions are attempted to the left and right of AY, with

[1]The ARPABET is the phonetic alphabet agreed upon by the ARPA Speech Understanding Research contractors.

```
                                MODIFIED
                                --------

---------------------AA----------------- - (AA)
                   - (AX) - ------DX---- - (IX)
                        (IX)-------------------IX-------------------(IX)


                                   TYPE
                                   ----

------T------(S)
                  (AE)-------AY-----(IY)
                                     (P)---------P---------


                                  SORT·BY
                                  -------

-----S------(S)
             -----OW----(UW)
                  (R) ----R-------
                               (D)-------D---(D)
                                (B)-----B-----(B)
                                             (AE)------AY-----(IY)
```

Figure 2-3. Coarticulation Example

different symbols representing the different nature of the articulation at the beginning and end. The third example, SORT BY, shows two interesting phoneme interactions. The R shares segments with OW as it would with any Front or Rounded vowel. Also, the T in SORT has been changed to a D to reflect the fact that the original T is unreleased, since it is followed by another plosive, B. Further, the system does not require plosive-plosive pairs to use time segments distinct from each other.

This Version A acoustic bottom end, though limited in its acoustic-phonetic capabilities, was accurate enough to demonstrate the use of the PLC concept. In the acoustic processing, the speech signal was sampled at the rate of 20,000 samples/second. Over every 10-millisecond interval, 18 parameters were abstracted from the 200 available speech samples for that interval. The number of parameters by category were:

5	frequency
3	frequency and amplitude
4	amplitude
3	amplitude and waveshape
3	waveshape

These parameters were input to a program that generated phoneme-label choice-lists for each 10-millisecond interval. Originally, a larger set of parameters were used. The choice was narrowed down to the 18 above using factor-analysis techniques. A number of speech samples were hand-marked with the ARPABET spelling. The hand-marked segment boundaries, along with the computed parameters, were fed to a series of programs that produced the labeling algorithms. To accomplish this, some results in distribution-free statistics were used. Included in the calculations were estimated occurrence probabilities for the individual phonemes. These were determined by a computerized analysis of the DMS grammar, results from the protocol experiments, and a few miscellaneous heuristics.

The use of prosodies by Version A is crude. Pause structure is looked for, but pitch contour is not. Pauses are handled by the exterior word boundary algorithms and treated as part of the adjacent word or phrase to which they belong (if known).

To date, 150 utterances have been passed through the Version A system. Generally, correct recognition scores are at the 60+% level. Errors of commission (complete but inaccurate recognition) have consistently been at the 10% level. The errors of commission generally happen when the wrong word choice was made at a particular spot and it was possible to complete the parsing meaningfully.

2.1.2 Acoustic-Phonetic Research

As mentioned earlier, a parallel effort was spent in developing a Version B of PLC-VDMS, which incorporates an acoustic bottom-end processor based on advanced acoustic-phonetic research. Substantial progress has been made in using this processor to segment continuous speech into recognizable phonemic units.

The incoming speech signal is digitized at 20,000 samples per second and saved in digitized form. The digitized signal is then passed through a digital-to-analog converter, and the resulting analog waveform is passed through three hardware bandpass filters whose ranges are 150-900 Hz, 900-2200 Hz, and 2200-5000 Hz, respectively. Two parameters are then extracted from each of the three resulting signals for each 10-millisecond segment: the maximum peak-to-peak amplitude and the zero-crossing count in the segment. The resulting six parameters (two from each of the three filtered signals) are used to assign a rough acoustic label to each segment. The following five labels are currently being used:

1) Vowels (VW)
2) Strident frications (SS)
3) Silence (SI)
4) Voiced consonants (VC)
5) Low-voiced or unvoiced areas (UV)

The next step in the processing is to refine these rough labels, i.e., impose on each segment a more accurate label than one of these five.

Within the classes VW and VC, more specific labels are assigned, using a vowel-recognition strategy based on speaker-dependent vowel-formant information [2], [3],[4]. This information is compared with the formants (obtained with the use of a Linear Predictive Coding--LPC--spectrum) at a single instant in time for each vowel to be identified. In an experiment designed to test this technique, four male subjects recited the Rainbow Passage, and the resulting speech was run through the system. Results showed that 91% of a total of 513 vowels in the four recitations of the Rainbow Passage were automatically located. Furthermore, the correct vowel was among the first three choices selected for 77% of those found; 65% were correctly identified by the first and second choices and 48% were correctly identified by the first choice alone (see Table 2-2).

TABLE 2-2. RESULTS OF VOWEL ISOLATION & RECOGNITION PROGRAM

	NUMBER OF VOWELS IN TRANSCRIPTION	PERCENT OF VOWELS FOUND	IDENTIFIED CORRECTLY		
			CHOICE 1 OR 2 OR 3	CHOICE 1 OR 2	CHOICE 1
OVERALL RESULTS	513	91%	77%	65%	48%
SPEAKERS:					
WAB	123	94%	84%	72%	59%
RDC	129	86%	72%	59%	40%
RVW	129	90%	76%	66%	47%
TCD	132	94%	75%	65%	48%

Since the time these results were originally reported, several improvements have been incorporated. The carrier sentences containing the vowels from which the vowel normalization tables are constructed have been modified, and we have modified the way diphthongs are handled in that the present routine breaks apart each component of the diphthong separately and then re-constructs it in a later part of the analysis process. These modifications have substantially increased the recognition capabilities of the vowel-normalization program.

Fricatives and plosives are characteristically found within sequences of segments labeled SS, VC, or UV. For these areas, a new technique developed at SDC, called the Low-Coefficient LPC (LCLPC), has been shown to provide meaningful spectra that correspond well with both acoustic-phonetic theory and with the experimental results of others [5]. Time resolution is sufficiently narrow to allow independent spectral analysis of the release, friction, and aspiration portions of an unvoiced plosive or to demonstrate spectral change within a consonant cluster, so that clusters such as /KS/ and /TS/ may often be distinguished. For analysis of unvoiced speech, the LCLPC uses the autocorrelation method with eight coefficients and a six-millisecond Hamming window, at a sampling rate of 20,000 samples per second. Analysis of spectra obtained in this way allows the following six classes to be distinguished:

1) Labial (LB)
2) Alveolar (AL)
3) Alveopalatal (AP)
4) Palatal and velar (PA)
5) Aspiration (AS)
6) Voiced or low energy (VS)

These classes correspond roughly to the spectral characteristics of unvoiced fricatives and plosives. Moreover, there is a correspondence between these classes and the articulatory positions of unvoiced fricatives and plosives. The classes LB, AL, AP, and PA ideally contain the following phonemes:

LB:	/P/, /F/, /TH/
AL:	/T/, /S/
AP:	/SH/
PA:	/K/

Although statistics are not yet available on the accuracy of this technique, it has been shown to be highly reliable in the analysis of well over 100 utterances by Version B of the speech understanding system.

Recent experiments have also confirmed that the glide /W/ and the liquid /L/ characteristically occur within the VW or VC classes. An innovative approach to recognizing these phonemes has been to augment a speaker's vowel formant table with the formant frequency values for /W/ and /L/. These formant values have consistently been easily distinguishable from the formants of the vowels, and have enabled us to accurately isolate and recognize /W/ and /L/. The glide /Y/ and the liquid /R/ are handled indirectly, again with the use of the speaker-dependent vowel formant table: if a 10-millisecond segment has been labeled /IY/, it is assumed that the segment could be a /Y/ with equal probability, and both labels are then assigned to the segment with the same probability. If a segment has been labeled /ER/ (again with the aid of the vowel table), the label /R/ is assigned to the same segment with equal probability.

Although we are not yet able to distinguish the various elements within the class of nasals, viz., /M/, /N/, /NX/, /EM/, /EN/, a single class name (NA) is used and has proved quite reliable. A segment is labeled NA if its first three formants (F_1, F_2, F_3) have the following characteristics:

1) $F_1 \leq \min(F_1) + 39$, where $\min(F_1)$ is the minimum F_1 value occurring in a speaker's vowel-formant table;
2) F_2 and/or F_3 is missing;
3) If (2) is not true, and one or more of the existing formants have wide bandwidths, for F_1, F_2, and F_3 the thresholds of the bandwidths are 200, 200, and 250 Hz, respectively.

All of the aforementioned segment-labeling procedures are used to construct an array of acoustic-phonetic data called the A-matrix. Each row of the A-matrix corresponds to a 10-millisecond segment of speech and contains a rough segment label (VW, SS, SI, VC, or UV), one or more refined segment labels and associated

```
PRINT CLASS WHERE DRAFT EQUALS FOUR
SESSION NO.=     1        SAMPLE NO.=   32        SUBJECT NO.=15605
SEG RS  1ST   2ND   3RD   4TH   5TH
 50 UV PA 72 AP 18 AL  3 LB  3
 51 UV LB 60 PA 30 AL  3 AP  3
 52 UV LB 47 PA 35 AP 11 AL  3
 53 SS AL 41 LB 31 AP 20 PA  3
 54 UV LB 60 PA 30 AL  3 AP  3
 55 VC AO 34 OW 32 AA 31  L 21 VS  1
 56 VC AO 34 OW 32 AA 31  L 21 VS  1
 57 VW OW 40 AA 34 UH 33
 58 VW AA 42 OW 37 AO 36
 59 VW OW 42 AO 38 AA 36
 60 VW AH 38 AA 35 AX 35
 61 VW AH 38 AA 35 AX 35
 62 VW AH 38 AA 35 AX 35
 63 VW AH 38 AA 35 AX 35
 64 VW AH 36 AE 36 AX 35
 65 VW AH 36 AE 36 AX 35
 66 VW AH 36 AE 36 AX 35
 67 VW AE 38 AX 36 AH 28
 68 VW AE 38 AX 36 AH 28
 69 VW AE 38 AX 36 AH 28
 70 VW AE 38 AX 36 AH 28
 71 VW AE 40 AX 34 AH 30
 72 VW AE 40 AX 34 AH 30
 73 VW AE 40 AX 34 AH 30
 74 VW AE 40 AX 34 AH 30
 75 VW AE 40 AX 34 AH 30
 76 VW AE 40 AX 34 AH 30
 77 VW AE 40 AX 38 AH 35
 78 VW AE 40 AX 38 AH 35
 79 VW AE 40 AX 38 AH 35
 80 VW AE 40 AX 38 AH 35
 81 VW AE 40 AX 38 AH 35
 82 VW AE 39 AH 38 AX 36
 83 VW AE 39 AH 38 AX 36
 84 VC AX 39 AH 38 AE 38 VS  1
 85 VC AH 40 AX 36 AE 29 VS  1
 86 VC AH 40 AX 36 AE 29 VS  1
 87 UV VS  1
 88 SS VS  1
 89 SS AL 60 AP 30 PA  3 LB  3
 90 SS AL 60 AP 30 PA  3 LB  3
 91 SS AL 53 AP 26 PA 13 LB  3
 92 SS AL 53 AP 26 PA 13 LB  3
 93 SS AL 53 AP 26 PA 13 LB  3
 94 SS AL 53 AP 26 PA 13 LB  3
 95 SS AL 53 AP 26 PA 13 LB  3
 96 SS AL 53 AP 26 PA 13 LB  3
 97 SS AL 53 AP 26 PA 13 LB  3
 98 SS AL 60 AP 30 PA  3 LB  3
```

Figure 2-4. A-Matrix Segment Label Choices for "Class"

probabilities based on the above procedures, formant frequency values, an estimate of fundamental frequency, RMS energy, and other acoustic-phonetic parameters used in the assignment of the phoneme and phoneme-class labels. In order to test the accuracy of the various labeling techniques, it is sometimes useful to extract from the entire A-matrix only those columns that contain the segment labels. An example of this reduced A-matrix is shown in Figure 2-4. The illustration shows 49 10-millisecond segments of the word "...class..." in "print class where draft equals four." The various column headings are:

SEG	- Segment number
RS	- Rough segment label (one of VW, SS, SI, VC, UV)
1ST, 2ND, 3RD, 4TH, 5TH	- First, second, third, fourth, and fifth label choices with associated probabilities

In segments 50 through 54 appear a sequence of feature labels characteristic of a /K/ going into an /L/. The /L/ itself appears in segments 55 and 56 with a score of 21. The vowel then appears in segment 57 and reaches its steady-state of /AE/ in segments 67 through 83. Finally, the AL feature label (characteristic of /S/) is seen in segments 89 through 98.

2.1.3 Version B Description

As mentioned earlier, the A-matrix forms the nucleus of the Version B ("Mini-System") acoustic-phonetic processing. The other major components of this system are:

1) The mini top end, which makes "next-word" predictions based upon syntax and some local semantic constraints;
2) The data base of information on U.S., U.S.S.R., and U.K. submarine fleets;
3) A lexicon, consisting of the phonemic spelling of the words in the data management language and the data base;
4) A lexicon program, in which phonemic variation rules have been implemented;
5) A mapping program that maps a phonemic string against the A-matrix.

The overall configuration of the mini-system is shown in Figure 2-5.

The format of the lexicon is an orthographic spelling of an entry accompanied by a string of one or more phonemic spellings. The spellings have a novel form, dictated by the requirement that rule application on the phonemic strings should be easy. Syllables are frequently the most appropriate units for rule application. Vowels are the nuclei of syllables, and the stress of the syllable is usually considered as residing with the vowel.

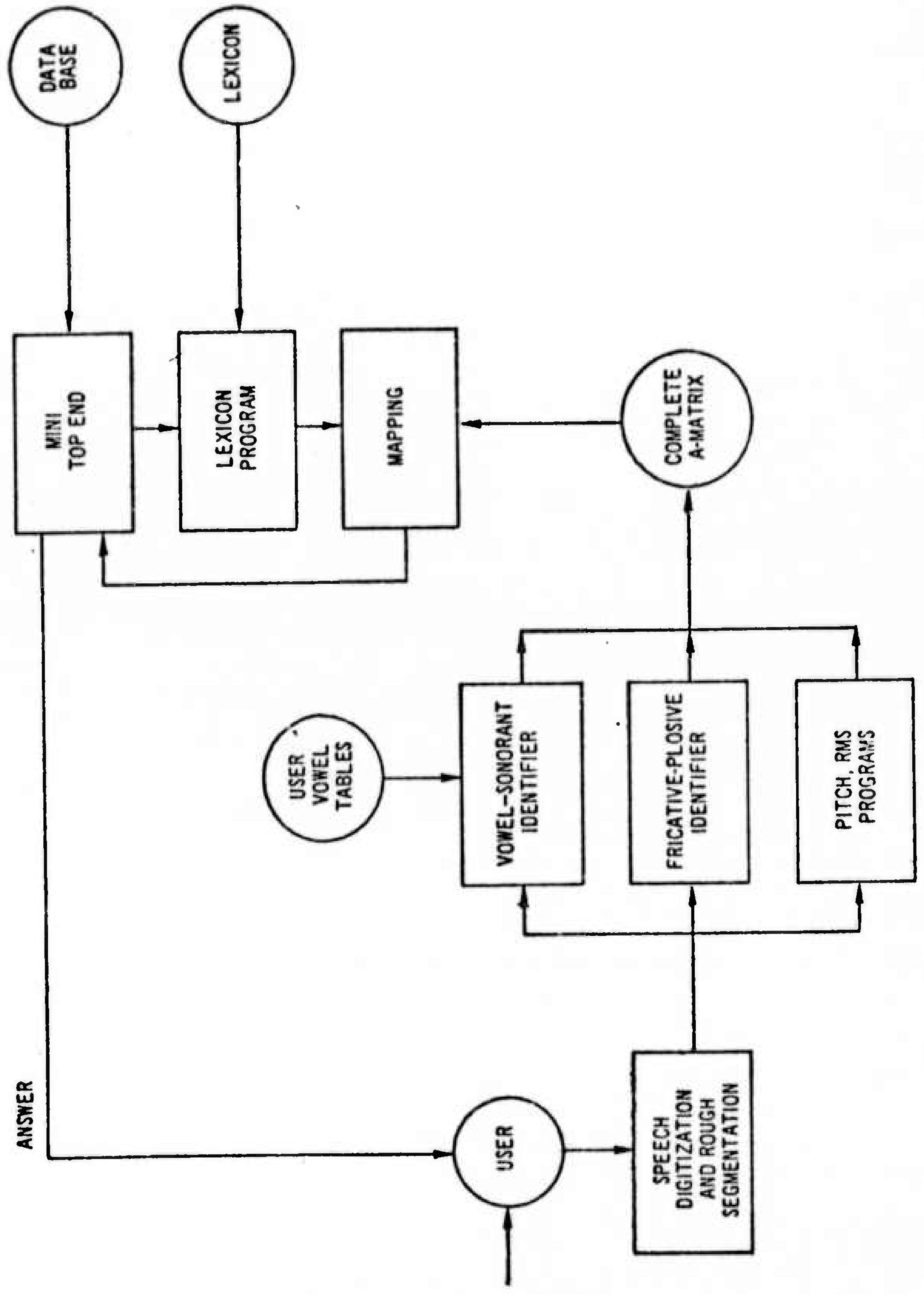


Figure 2-5. Mini-System Configuration (Version B)

Quantit, variant #2: /kwánədi/

1)
$$\begin{tikzcd}
 & \text{Stress: 2} \\
\text{AA} \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & \text{Init cons: K W} \\
 & \text{Fin cons: N} \\
 & \text{Last: (no)}
\end{tikzcd}$$

2)
$$\begin{tikzcd}
 & \text{Stress: 0} \\
\text{AX} \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & \text{Init cons: (none)} \\
 & \text{Fin cons: (none)} \\
 & \text{Last: (no)}
\end{tikzcd}$$

3)
$$\begin{tikzcd}
 & \text{Stress: 1} \\
\text{IY} \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & \text{Init cons: D} \\
 & \text{Fin cons: (none)} \\
 & \text{Last: (yes)}
\end{tikzcd}$$

(A rule will produce /kwánədɪ/ also; i.e.,
"IY with 1 stress alternates with IH".)

Figure 2-6. The Structure of Lexical Entries

A single phonemically encoded word consists of a sequence of vowels, each one with an associated set of properties. The vowel, together with its properties, comprises a syllable. The properties are (1) stress, (2) a preceding consonant cluster (possibly null), (3) a following consonant cluster (possibly null), and (4) an indication as to whether this syllable is word final. Figure 2-6 is an illustration of this design. It is a relatively straightforward task to assemble the syllables into a phonemic string, and this is done after the rules are applied.

The mapping program [6] maps a phonemic string onto a computer representation of an interval of continuous speech. The procedure has as its inputs a hypothesized phonemic string and a matrix of labels and probabilities obtained from the A-matrix. These two inputs are mapped by an interactive technique, which (1) searches for individual phonemes in an appropriate order, not necessarily left-to-right; (2) continuously adjusts for phoneme duration variations; and (3) can deal with intraword coarticulation effects. The mapping procedure yields estimated time boundaries and a goodness-of-match score for the hypothesized phoneme string. This algorithm performs automatic recognition of words and phrases within Version B.

The following 19 test utterances were run through Version B:

DESCRIBE.

EXPLAIN SHOW.

REPEAT WHERE CLASS EQUALS CRUISE MISSILE.

PRINT ENTRIES WHERE SAME.

PRINT BEAM WHERE CLASS EQUALS PATROL.

TOTAL MISSILES WHERE SAME.

* PRINT DRAFT WHERE SAME.

COUNT WHERE SAME.

EXPLAIN EXPLAIN.

* SHOW COUNTRY.

* REPEAT WHERE TYPE EQUALS TANG.

* PRINT CLASS WHERE DRAFT EQUALS NINE.

SHOW TYPE.

TOTAL BEAM WHERE SAME.

* TOTAL TORPEDO TUBES WHERE SAME.

REPEAT WHERE CLASS EQUALS ATTACK.

REPEAT WHERE CLASS EQUALS OCEAN GOING.

REPEAT WHERE CLASS EQUALS PATROL.

* COUNT WHERE TORPEDO TUBES EQUALS ONE.

---

*Missed completely or partially by the system

Thirteen of these 19 utterances were comprehended correctly. It is important to note that this score was achieved without the aid of a thematic memory and virtually without phonological rules. Some local semantic constraints were used, however, to aid the parsing.

2.1.4 Related Research

Although the majority of the research over the previous contract year consisted of tasks directly related to the development of Versions A and B of a speech understanding system, other related studies were conducted. Two such studies are particularly worthy of note:

1) A protocol experiment with a simulated vocal data management system.

2) An experiment to test the effects of pre-emphasis on the formant analysis of speech using linear prediction techniques.

The purpose of the protocol experiment was twofold: (1) to study the way in which thematic patterns relate to a user's goal-directed behavior, and (2) to study the relation between pause structure and changes in fundamental frequency and the types of commands given to the system by the user. In this experiment, each subject was given a problem telling him to extract information from the data base using the VDMS query language. He spoke his spontaneous commands into a microphone inside the laboratory's sound booth, and his commands were heard over headphones by a receiver outside the booth. As the receiver heard the commands, he acted like the speech "recognizer" and typed the message to the data management system on the Raytheon 704. The response was sent directly to the subject in the booth. Even though the language and data base are constrained in the context of data management, some interesting and useful results about prosodics and thematics have been derived [7], [8].

For the pre-emphasis experiment, two utterances were selected: one completely voiced and the other containing both voiced and unvoiced segments. A high-quality analog tape recording was made of the utterances as spoken by both a male and a female speaker. Each of the resulting four analog recordings was then digitized directly and saved. The same analog recording was also subjected to analog pre-emphasis while being digitized and saved. Finally, the non-pre-emphasized digitized recording was then differentiated and saved.

Twelve digital recordings were thus created, and each was run through an LPC spectrum calculation and peak-picking procedure. The frequencies at which the peaks occurred were then plotted against time. With the aid of sonographs of the utterances, it was possible to connect the various frequencies and create reasonable formant trajectories in each case. Upon comparison of the resulting formant plots, it was shown that:

1) For male speakers, there is very little discrepancy in the results of analyzing pre-emphasized and non-pre-emphasized speech.
2) For female speakers, pre-emphasis should be used, since it leads to more accurate formant estimates.
3) Differentiating the waveform provides results as good as analog pre-emphasis.

The results of this experiment were also helpful in the design of our acoustic processing configuration.

2.2 SUMMARY

Over the previous contract year, two complete speech understanding systems were built: Version A, with a sophisticated syntactic and semantic top driver and limited acoustic-phonetic support, and Version B, which contains all of the acoustic-phonetic processing algorithms for vowels, fricatives, plosives, liquids, glides, and various phoneme classes, but has limited syntactic and semantic support. The first major task to be accomplished during the next year will be the construction of a speech understanding system that incorporates the top driver of Version A and the acoustic-phonetic processing of Version B. Indeed, substantial progress has already been made toward the completion of this goal.

The resulting system will be tested with our current 150-word vocabulary for two or three male speakers and one female speaker. After the testing, various ways of loosening the data management system grammar will be attempted. The grammar will be loosened with the present 150-word vocabulary. Once a sufficiently high recognition rate has been achieved, the vocabulary will be extended to about 500-600 words. In parallel with these efforts, we will use the results of further protocol experiments to improve the thematic patterning and prosodies of the thematic memory. A reliable fundamental frequency extraction program has been implemented in the A-matrix processing, so that results about prosodies derived from the protocols will be usable in the system.

A major portion of research will be devoted to extending the accuracy of our acoustic-phonetic processing. As a first step towards this goal, we will complete the refinement of a formant-tracking program that was recently implemented. It is expected that formant tracking over voiced areas of speech may eventually replace the information obtained from the front-end filters and will provide more accurate frequency and amplitude information to assist us in the study of vowels, nasals, liquids, and glides. Moreover, additional potential exists in the method we have developed for fricative and plosive analysis. Specifically, we believe that its handling of voiced plosives and fricatives can be significantly improved and that we will soon be able to meaningfully distinguish between voiced and unvoiced plosives.

2.3 BIBLIOGRAPHY

[1] Barnett, J., "A Vocal Data Management System," IEEE Transactions on Audio and Electroacoustics, Vol. AU-21, No. 3 (pp. 185-188), June, 1973.

[2] Kameny, I. and Weeks, R. V., "A Vowel Isolation and Recognition Strategy Based on Speaker-Dependent Vowel Tables," ARPA SUR Note #89 (NIC #17516), July 5, 1973.

[3] Kameny, I. and Weeks, R. V., "An Experiment in Automatic Isolation and Identification of Vowels in Continuous Speech," presented at the 86th meeting of the Acoustical Society of America, Los Angeles, October, 1973.

[4] Kameny, I. and Weeks, R. V., "Comparison of Speaker-Dependent Vowel Tables with Peterson-Barney Vowel Tables in the Selection of Vowel Label Choices," ARPA SUR Note #98 (NIC #17927), July 30, 1973.

[5] Molho, L., "Automatic Recognition of Fricatives and Plosives in Continuous Speech using a Linear Prediction Method," presented at the 86th meeting of the Acoustical Society of America, Los Angeles, October, 1973.

[6] Gillmann, R., "Automatic Verification of Hypothesized Phoneme Strings in Continuous Speech," presented at the 86th meeting of the Acoustical Society of America, Los Angeles, October, 1973.

[7] Diller, T. C., "Prosodies Appearing in the SDC Vocal Data Management Dialogues," ARPA SUR Note #93 (NIC #17656), July 10, 1973.

[8] Diller, T. C., "Thematic Patterning in the SDC Vocal Data Management Dialogues," ARPA SUR Note #94 (NIC #17663), July 20, 1973.

2.4 PROJECT PERSONNEL

H. B. Ritea, Project Leader
J. Barnett
W. Brackenridge
R. Gillmann
I. Kameny
P. Ladefoged (consultant)
L. Molho
D. Pintar
R. Weeks

3. DEDUCTIVE INFERENCE RESEARCH

One of the obvious deficiencies of English Q/A systems has been their inability to draw inferences to answer queries posed by the average user of such systems. The CONVERSE Project's sophisticated Q/A system provided a proper vehicle and environment within which to install the requisite inference making capability.

Much has been done toward mechanizing deduction over the past 15 years. The work of Robinson [1] and others in mathematical deduction, the procedural deducers such as MIT's PLANNER [2] and CONNIVER [3] and SRI's QA4 [4], and the work of Elliott[5] and others on special-purpose deduction exemplify the present state of the art.

The goal of our current research is to design and implement a deductive processor that will support practical question-answering in realistic environments involving large amounts of both general and specific information. To meet this objective we have adopted, extended, and integrated techniques from all three of the previous approaches to mechanized deduction: mathematical, procedural, and special-purpose. Secondly, we have focused on one difficult problem that remains unsolved in all of the previous approaches, the problem of selecting from a large file of facts and general assertions the very few that are relevant for a particular required deduction. Thirdly, considerable attention is being devoted to optimizing the interaction of deductive components in a deductive processing module and to looking at the problems of effectively interfacing this module with language processing routines in one direction and data management routines in the other.

3.1 PROGRESS

Figure 3-1 illustrates the major components of SDC's deductive data management system:

1) A relational data management module based on the CONVERSE system, designed to maintain, update, and retrieve from a body of specific factual information organized principally around relationships of central interest to a user community.

2) A language processor that accepts a user's queries, facts, or general assertions and translates these into formal intermediate language expressions that precisely specify and quantify the relationships of interest.

3) A deductive processing module that uses a file of general assertions (possible premises) to generate deductive structures whose logical consequence is the information that is desired but that is not stored explicitly in the system's files. (The addition of a deductive processor augments the capabilities of the language

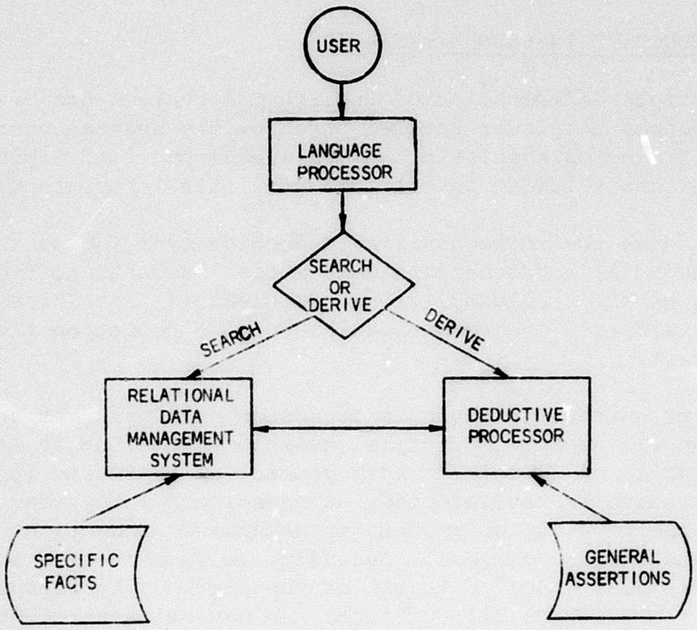


Figure 3-1. A Deductive Data Management System

processor by extending the vocabulary and the range of expression forms available to a user. It also improves the performance of the relational data management module.) The deductive processor will facilitate acquisition of the information needed in its files of general assertions. When a user introduces a new relation, the processor will engage in a short dialog with the user to elicit the relevant structural aspects of the relations. This will ease the user's task of remembering and entering required and useful general information about the new relation. It will not be necessary for the user to precisely formalize this general information (a task roughly comparable to writing a computer program) before he can enter it into the system.

4) A decision module that decides, for each intermediate language expression, whether it should be sent to the relational data management system, for search of the file of specific facts, or to the deductive processor, for the utilization of the file of general assertions. If the decision module ascertains that there is a relatively complete file of instances for each relation involved in the query, or that the instances are computable with available special-purpose routines, then the query will be sent to the relational data management system for answer by look-up and computation. When the query involves relations for which specific instances are not computable or are only very incompletely stored, if at all, then the query is sent to the deductive processor.

Modules of type (1) and (2) have already been constructed and used in the CONVERSE and other English question-answering systems. Our research during the past year has focused on the design and implementation of type (3) and type (4) modules and of a language in which to express inferential relationships economically.

3.1.1 The Intermediate Language

A key design requirement is a precise but expressively powerful language interface between the deduction module on the one hand and the language-processing and data-management modules on the other. An intermediate language (IL2) has been designed that is rich enough to represent all of the kinds of information that need to be conveyed to the deductive processor and yet simple and straightforward enough that it does not pose difficulties in interfacing with a variety of data base management systems and front-end language processors.

Four primary design goals have determined IL2:

1) To create a language with the full expressive capability of the predicate calculus, thus giving it sufficient generality to drive both data management systems and deductive processors.

2) To create a language in which operations on sets and relations can be represented easily and without the restrictions, found in classical relational calculi, to binary relations.

3) To create a language that directly reflects the meaning of English sentences analyzed in terms of deep structures based on case relations and that uses variables only to the extent that their use directly reflects English anaphora.

4) To create a language in which the selection and permutation of argument terms (variables and constants) is to be explicitly indicated rather than just pictured (as in most logical languages like the predicate calculus).

The goals derive from the fact that IL2 serves the dual role of being both a target language enabling precise representation of the meaning of English statements and a data-management and deduction-driver language capable of controlling the storage and processing of relational data and general data from which inferences are to be drawn.

Representative Statements in IL2 have the general form:

(Header, Selection conditions, Output operations)

The Header identifies sets, relations and functions (as well as variables to be used for naming argument places or cases of the relations), and the Selection conditions specify conditions that must be satisfied by members of these sets or relations before they are subjected to the specified Output operations.

Figure 3-2 illustrates the use of IL2 in three applications. Figure 3-2a illustrates the use of IL2 to represent the meaning of four English copulative sentences requiring the handling of identity, set membership, set inclusion, and predicate adjectives, respectively. As can be seen, each expression consists of the equality relation (EQ) as header, followed by two selection conditions. Each selection condition (and output operation) clause is set off by parentheses and every clause begins with one of a small set of descriptive ($\alpha,\eta,\text{ɿ}$), interrogative ($), and selective ($=,\neq,>,<$,MAX,MIN) operations. In Figure 3-2a, these qualifying operations are immediately followed by the name of the qualified entity or concept.

Descriptive operators are an important element of the language. In Figure 3-2a, the "ɿ" symbol stands for the definite description operator that functions like "the" in English. The first symbolization therefore states that the entity "Scott" is formally identical with the entity that happens to be the author of Waverly. In a similar manner, the "η" symbol represents an indefinite description operation (it functions like "a" or "some" in English) and is here used to represent indefinite members of the classes of celestial bodies, political subdivisions, and bright entities. Finally, the "α" symbol is introduced for the general description operation (it functions like "every" or "all" in English).

The IL2 descriptive operators are used in a different application in Figure 3-2b, where the sentence "Every ship sails some sea" is compactly and conveniently symbolized as contrasted with the equivalent predicate calculus symbolization: $\forall x(\text{SHIP}(x) \rightarrow \exists y(\text{SEA}(y)\&\text{SAIL}(x,y)))$. Also illustrated is the use of the interrogative operator ($) in calling for output and the restriction of subrelations (cities farther east). The symbols "D", "N", "S", and "G" stand for the syntactic case relations domain, neutral, source, and goal, respectively. They form an integral (though optional) part of IL2 and are used to associate conceptual arguments with their header relation.

Figure 3-2c illustrates the range of standard set-theoretical operations representable in IL2. IL2 is translatable by machine into either predicate calculus expressions or calls on data base retrieval routines. Within IL2, one can formulate arbitrarily complex computable functions in a LISP-like form to specify procedural-semantic advice or special-purpose deductive strategies.

SCOTT IS THE AUTHOR OF WAVERLY

[EQ (= SCOTT (℩ AUTHOR OF WAVERLY)]

THE SUN IS A CELESTIAL BODY

[EQ (℩ SUN) (η CELESTIAL BODY)]

EVERY CITY IS A POLITICAL SUBDIVISION

[EQ (α CITY) (η POLITICAL SUBDIVISION)]

THE SUN IS BRIGHT

[EQ (℩ SUN) (η BRIGHT)]

a. Copulative Sentences

EVERY SHIP SAILS SOME SEA.

[SAIL (α D SHIP) (η N SEA)]

WHAT SHIPS SAIL EVERY SEA?

[SAIL ($ D SHIP) (α N SEA)]

WHAT FLIGHTS DEPART FROM CITIES FOR CITIES FURTHER EAST?

[DEPART ($ D FLIGHT) (= S CITY) (= G CITY) (= (G S) EAST)]

b. Verbal Sentences

DOMAIN	$\{X : (\exists Y)(XSY)\}$	[S ($ D)]
DOMAIN RESTRICTION	$\{\langle X, Y\rangle : XSY \wedge X \in A\}$	[S (= D A)]
IMAGE	$\{Y : (\exists X)(XSY \wedge X \in A)\}$	[S (= D A) ($ N)]
COSET	$\{Y : a\, S\, Y\}$	[S (= D a) ($ N)]
COMPOSITION	$\{\langle X, Z\rangle : (\exists Y)(XQY \wedge YSZ)\}$	[(S Q) (= C_2 C_3) ($ C_1 C_4)]

c. Set-Theoretic Operations

Figure 3-2. Intermediate Language Symbolizations

3.1.2 Deductive Files and Processors

Figure 3-3 illustrates the principal files (rounded boxes) and processors (square boxes) that presently constitute the deductive processor. They are discussed below.

3.1.2.1 Files

The four principal files are: (1) the general assertion file, (2) the predicate connection graph, (3) the semantic advice file, and (4) the variable substitution file. This section discusses how these files are related to each other and how the varying kinds of information they contain are needed at different steps in the deductive inference process.

As was shown in Figure 3-1, the deductive processor has access to a file of general assertions, or premises. These premises are represented in Skolemized, quantifier-free form, as in resolution theorem provers. However, instead of resolution's conjunctive normal form, a primitive conditional form is used as the basic type of structure. Primitive conditionals are centrally connected by the implication sign operating on groupings of literals combined conjunctively or disjunctively (see Figure 3-4). Each literal is an atomic formula or a negated atomic formula. The primitive conditional form is rich enough to represent anything expressible in the first-order predicate calculus. It has been chosen principally because it facilitates finding chains of deductively linked middle-term predicates. It also facilitates the storage of information in such a way that the strategic or heuristic implications of the original formulation are not lost in the system.

The predicate connection graph is abstracted from the information available in the premises. Nodes represent predicate occurrences. Each edge between a pair of nodes represents a possible deductive interaction between the predicate occurrences represented by the nodes. To guide and constrain searches for possible deduction paths, many different kinds of information are associated with the nodes and arcs. The predicate connection graph is of key importance in the system and is one of its unique features. It represents explicitly and compactly a great deal of detailed structural information about general assertions and their possible deductive interconnections that can be quickly accessed and scanned.

Figure 3-4 illustrates a series of premises and their representations in primitive conditional forms; Figure 3-5 shows the resulting predicate connection graph. The solid lines in Figure 3-5 picture u-arcs (unification arcs), which represent deductive interactions between predicate occurrences in different premises. (In a resolution system, these arcs are recomputed for every required new deduction.) Since establishing large numbers of these arcs can be time consuming, they are computed only once, at the time a premise is entered into the system, and then filed. Another kind of information in the predicate connection graph is the deductive dependency link, which represents deductive dependency between predicate occurrences within a single premise. Only two types of dependency links are shown in Figure 3-5: I (implication) links and C (conjunction) links.

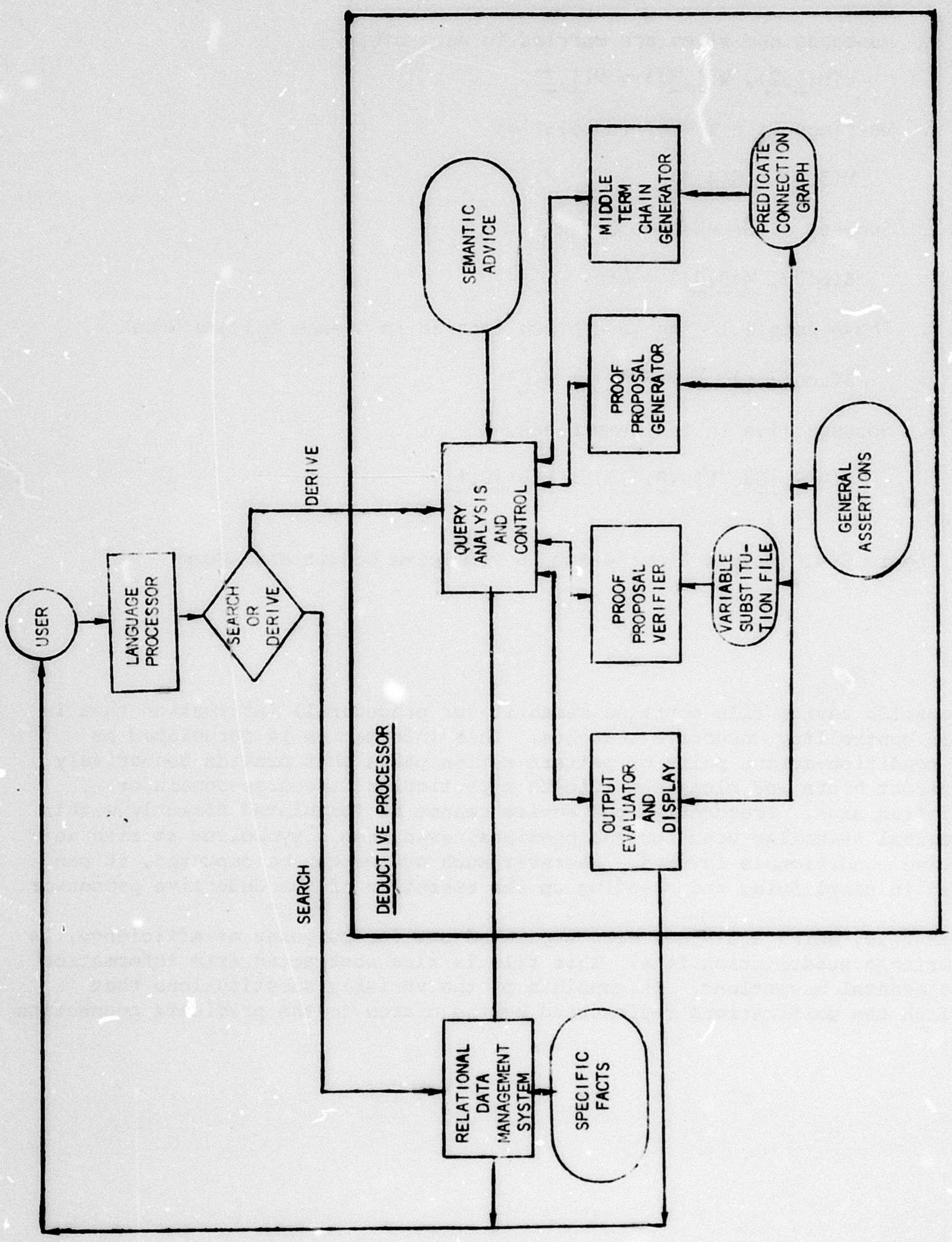

USER
LANGUAGE PROCESSOR
SEARCH OR DERIVE
DERIVE
SEARCH
DEDUCTIVE PROCESSOR
SEMANTIC ADVICE
MIDDLE TERM CHAIN GENERATOR
PREDICATE CONNECTION GRAPH
PROOF PROPOSAL GENERATOR
QUERY ANALYSIS AND CONTROL
GENERAL ASSERTIONS
PROOF PROPOSAL VERIFIER
VARIABLE SUBSTITU-TION FILE
OUTPUT EVALUATOR AND DISPLAY
RELATIONAL DATA MANAGEMENT SYSTEM
SPECIFIC FACTS

Figure 3-3. Deductive Files and Processors

1. Husbands and wives are married to each other

   $v(H(\underline{1},\underline{2}), W(\underline{1},\underline{2})) \supset M(\underline{1},\underline{2})$

2. Marriage is a symmetric relation

   $M(\underline{3},\underline{4}) \supset M(\underline{4},\underline{3})$

3. Spouses of Greeks are Greek

   $\&(G(\underline{5}), M(\underline{5},\underline{6})) \supset G(6)$

4. Those people living in a place located in Greece ($\underline{g}$) are Greek

   $\&(Loc(\underline{7},\underline{g}), Liv(\underline{8},\underline{7})) \supset G(\underline{8})$

5. Spouses live in the same place

   $\&(M(\underline{9},\underline{10}), Liv(\underline{9},\underline{11})) \supset Liv(\underline{10},\underline{11})$

Figure 3-4. Sample Premise Set in Primitive Conditional Form

The semantic advice file contains semantic (or procedural) information that is used in controlling deductive searches. This information is formulated as basic condition-action pairs or pattern-action pairs that provide deductively significant hints and clues specific to a particular discourse domain or application area. Frequently, the advice cannot be formulated directly within the logical symbolism used for the premises, even when a symbolism as rich as primitive conditionals is used. Wherever such advice can be captured, it can be used in simplifying and speeding up the operation of the deductive processor.

Another file, which again has been separated out for purposes of efficiency, is the variable substitution file. This file is also abstracted from information in the general assertions. It consists of the variable substitutions that establish the unifications represented by the u-arcs in the predicate connection graph.

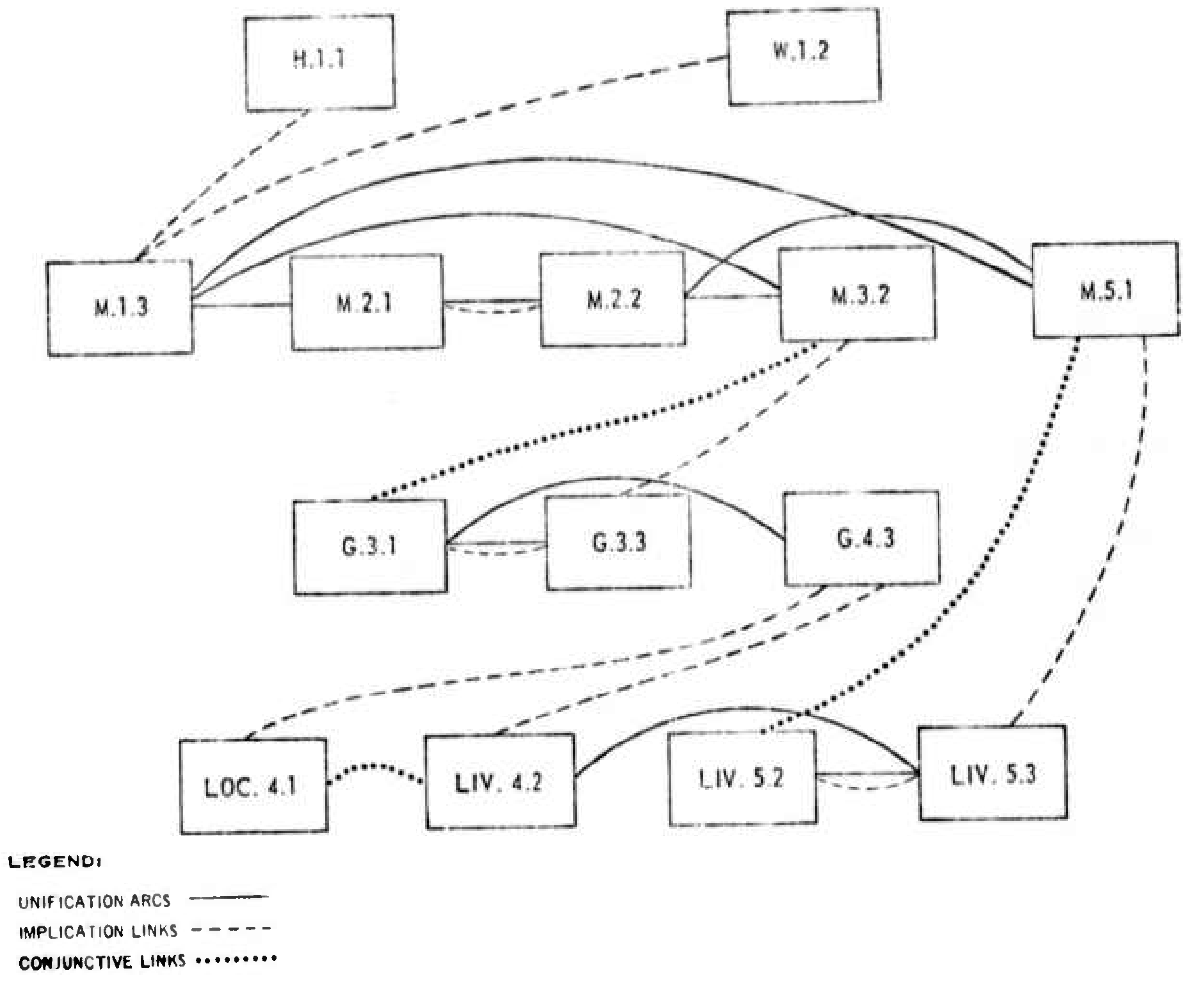


Figure 3-5. Predicate Connection Graph for the Sample Premise Set

3.1.2.2 Processors

The processors involved in the deductive system are briefly discussed below. (For a more detailed description, see Travis, et al. [7], which is, however, out of date with respect to several aspects of the system organization.)

Control

The control processor accepts IL2 expressions and generates appropriate responses. Typically, the control processor will call the deductive processors as a series of co-routines to be scheduled, operated, and controlled as dictated by the nature of the query and the detailed content of the semantic advice, general assertion, and data base files. For example, if semantic advice is available for a problem, the control processor may decide to generate a single proof plan and proceed directly to verification and instantiation (data base retrieval). In other situations a number of proof plans may be generated first, and the control processor may call the proof proposal verifier to aid in selecting the most promising plan before invoking the data-base retriever. This pseudo-parallel operation will be adapted from the Control Structure Language developed and implemented by SDC's Speech Understanding Research project. (See Section 2 for a description of the speech work.)

Middle-Term-Chain Generator

The predicate connection graph is used to find chains of middle-term predicates that deductively link the assumption and goal predicates extracted from an IL2 query. The basic proof strategies of natural deduction, reductio ad absurdum, and proof-by-cases are automatically incorporated into the predicate chain generator. The chain generation process may be visualized as one in which a series of expanding "wave fronts" is generated from each assumption and goal predicate. These wave fronts represent deductively significant possible paths from each predicate. As the wave fronts expand from assumption predicates toward goal predicates, intersections are taken in order to determine when an assumption wave front impinges upon a goal wave front--an occurrence that indicates the beginning of a promising proof plan.

Proof-Proposal Generator

The proof-proposal generator combines chains and premises into skeletal proof plans, possibly using ready-made skeletons suggested by second-order structural properties of the predicates involved. These plans represent structures that can be turned into complete proofs upon discovery of usable instantiations (definite values for individual variables) and upon verification that these instantiations do not conflict with each other. These plans give direction to the verification process charged with constructing full deductive detail and prevent the large amount of instantiation detail work that terminates at deadends in other approaches to mechanized deduction.

Proof-Proposal Verifier

For a single problem, the proof-proposal generator may produce several proposals that are passed on for verification. The proposals will be ranked according to the complexity of their variable structures; this ranking is used as indicative of likelihood of success or failure. For each proposal, the variables and their substitutions in the proof structure (variable flow) will be examined to determine whether there are blockages (variables taking on conflicting values).

Output Evaluator and Display Generator

If the system completes a successful verification and instantiation of a proof proposal, it will output direct answers and the derivations on which they are based. When derivations are not complete, the system will output either conditional answers or partial derivations that, in many cases, will provide clues to missing information that the user may be able to acquire from some other source. Each system response will be accompanied by an intelligible display of the logical argument on which it is based.

3.1.2.3 A Sample Derivation

The example in Figure 3-6 illustrates the use of existential quantifiers and the resulting Skolem functions (in this example, f($\underline{1}$,$\underline{2}$) and g($\underline{1}$,$\underline{2}$)). Such a representation gives the deductive system a much more general capability than systems that do not allow existentially quantified variables (e.g., MICRO-PLANNER [6]).

The deductive processor has been designed to handle the Skolem functions automatically derived from existential quantifiers in an efficient manner. Along with negation, conjunction, and disjunction on both sides of the implication sign, this ensures the full generality of a first-order predicate calculus. Techniques for efficiently utilizing second-order properties and provisions for handling higher-order constructs when and if they are desired are being built into the system. Far from overburdening the system with too much generality at the expense of performance, we believe that higher-order constructs will enable us to carry out certain deductive steps far more efficiently than they could be in more limited systems.

3.1.3 Use of Second-Order Properties

The work of Elliott on special-purpose deduction was mentioned earlier. His system was based on the use of second-order structural properties like transitivity and symmetry. These structural properties were used to direct the search for relevant specific facts (relation instances) in a relational data base. These techniques have been adapted and extended for SDC's general inference system.

Facts: S(a,b) -- "a" is sibling of "b"

C(b,c) -- "b" is cousin of "c"

Premises:

(1) A cousin is an offspring of a sibling of a parent.

∀x∀y(C(x,y) <=> ∃w∃z(O(x,w)&S(w,z)&P(z,y)))

This divides into two premises shown below as primitive conditionals using numbers as variables:

(1a) C(1,2) => &(O(1,f(1,2)),S(f(1,2),g(1,2)),P(g(1,2),2))

(1b) &(O(5,7),S(7,8),P(8,6)) => C(5,6)

(2) A sibling of an offspring is an offspring.

∀x∀y∀z(O(x,y)&S(z,x) => O(z,y))

In primitive conditional form:

&(O(9,10),S(11,9)) => O(11,10)

Question: Is "a" a cousin of "c"?

SUCCESSFUL DERIVATION:

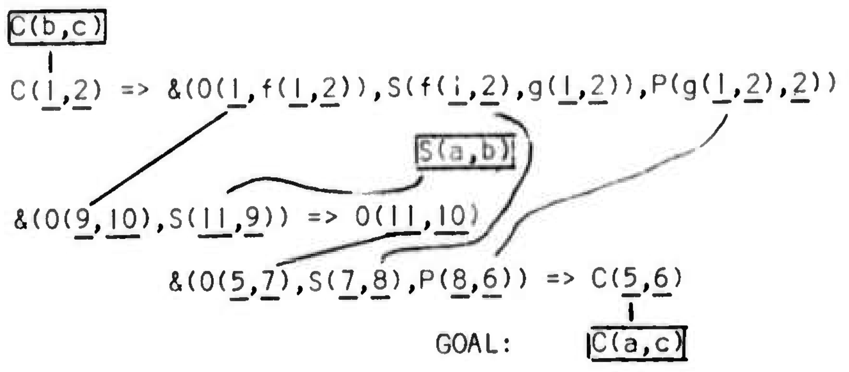

C(b,c)
C(1,2) => &(O(1,f(1,2)),S(f(1,2),g(1,2)),P(g(1,2),2))
S(a,b)
&(O(9,10),S(11,9)) => O(11,10)
&(O(5,7),S(7,8),P(8,6)) => C(5,6)
GOAL: C(a,c)

Figure 3-6. A Sample Derivation Using Existential Quantifiers

The nine second-order structural properties used by Elliott are illustrated in Figure 3-7. Of Elliott's nine properties, there are 256 possible logical combinations. He showed that of these 256, only 32 are deductively productive. Each one of the 32 can be identified with a subroutine for appropriately searching predicate extensions--or (extending Elliott) for inserting a ready-made skeleton at an appropriate place in a deduction plan.

The appropriate use of Elliott-like techniques and their extensions simplifies the generation and evaluation of proof plans. This is done by "factoring out" the complicating details due to second-order concepts that would otherwise be represented. These factored-out constructs do nothing to limit generality but do provide a means for more intelligently and efficiently searching for suitable instantiations by use of higher-level proof plans.

Figure 3-8 shows an elementary dialog designed to illustrate how, by answering two simple questions, a user can get into the system the information that the North of relation is transitive and symmetric. The figure also illustrates part of the system's "understanding" of North of in simple contexts derived from the above mentioned elementary user-machine dialog. The system will be able to reject anomalous inputs (such as redundant and inconsistent facts), accept definitions of converse relationships, answer questions correctly on the the basis of stored facts, and know when to respond "I don't know."

3.2 SUMMARY

Two of the major program modules of the deductive processor have been programmed in LISP: The middle-term chain generator and the proof proposal generator. The first version of the chain generator has been modified to interact with the proof proposal generator and to operate from the structure for the predicate connection graph. An initial experimental corpus has been constructed for use in program check-out; it presently consists of about 50 general assertions on kinship relations. Deductive links and variable substitutions for this corpus have been constructed manually, and these data have been keypunched to create the predicate connection graph, premise file, and variable substitution file. The logic for the file-maintenance routines necessary to create all system files from an input set of general assertions has been flowcharted. We have also created a decision table for the logic necessary to elicit the structural meanings of relations along the lines discussed in the previous section. Experimental production of proof plans is beginning.

REFLEXIVE	A relation R is reflexive if whenever aRb is true then both aRa and bRb are true. Example: is equal to.
IRREFLEXIVE	A relation R is irreflexive if aRa is always false. Example: is greater than.
SYMMETRIC	A relation R is symmetric if whenever aRb is true then bRa is also true. Example: is equal to.
ASYMMETRIC	A relation R is asymmetric if whenever aRb is true then bRa is false. Example: is north of.
TRANSITIVE	A relation R is transitive if whenever aRb and bRc are both true then aRc is also true. Example: is north of.
1-LEADER	A relation R has 1-leader if for any given object there is only one subject. Example: is the mother of.
1-FOLLOWER	A relation R is 1-follower if for any given subject there is only one object. Example: weighs.
NOREGROWTH	A relation R has property noregrowth if whenever aRb is true then it is not true that there exists a chain aRz_1, $z_1Rz_2, \ldots, z_nRb$ which are all true. Example: is the son of.
UNLOOPED	A relation R is unlooped if whenever aRb is true then there does not exist a collection bRz_1, $z_1Rz_2, \ldots, z_nRa$ which are all true. Example: is north of.

Figure 3-7. Second-Order Properties of Relations

1. # Beverly Hills is north of Downey.
2. I don't understand <u>north of</u>. Suppose one thing is north of some second thing that in turn is north of some third thing. Will it then be the case that the first is north of the third?
3. # yes.
4. OK, if one thing is north of a second thing, will it always be the case that the second is also north of the first?
5. # no.
6. OK, will it ever be the case?
7. # no.
8. OK, I understand Beverly Hills is north of Downey.
9. # Beverly Hills is north of Long Beach.
10. OK, # Inglewood is north of Long Beach.
11. OK, # Canoga Park is north of Canoga Park.
12. This is not possible, something cannot be north of itself.
13. # Canoga Park is north of Beverly Hills.
14. OK, # Canoga Park is north of Long Beach.
15. Not stored-already known.
16. # South of is the converse of north of.
17. OK, # Santa Monica is south of Canoga Park.
18. OK, # Downey is north of Canoga Park.
19. This is incorrect, not stored.
20. # Is Inglewood north of Beverly Hills?
21. No.
22. # Is Downey south of Inglewood?
23. I don't know.
24. Is Canoga Park north of Van Nuys?
25. I don't understand "Van Nuys."
26. # What is south of Canoga Park?
27. Santa Monica, Beverly Hills, Downey, Inglewood, and Long Beach.
28. # What is north of Downey?
29. Beverly Hills and Canoga Park.

Figure 3-8. Example of User-Computer Dialog in Eliciting and Using Second-Order Properties of Relations

3.3 BIBLIOGRAPHY

[1] Robinson, J. A., "A Machine-Oriented Logic Based on the Resolution Principle," Journal of the ACM, Vol. 12, No. 1 (1965), pp. 23-41.

[2] Hewitt, C., "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot," MIT Report A.I. TR-258, Cambridge, 1972.

[3] McDermott, D. V., and Sussman, G. J., "The CONNIVER Reference Manual," MIT A.I. Memo 259, Cambridge, 1972.

[4] Rulifson, J. F., Derksen, J. A., and Waldinger, R. J., "QA4: A Procedural Calculus for Intuitive Reasoning," Stanford Research Institute A.I. Group Tech. Note 73, Menlo Park, 1972.

[5] Elliott, R. W., "A Model for a Fact Retrieval System," University of Texas Computation Center Report TNN-42, Austin, 1965.

[6] Sussman, G., Winograd, T., and Chalniak, E., "Micro-PLANNER Reference Manual," MIT A.I. Memo 203, Cambridge, 1970.

[7] Travis, L., Kellogg, C., and Klahr, P., "Inferential Question-Answering: Extending CONVERSE," System Development Corporation Report SP-3679, Santa Monica, 1973.

3.4 PROJECT PERSONNEL

C. H. Kellogg, Project Leader
P. Klahr
L. Travis (consultant)

4. IMAGE AND SIGNAL PROCESSING

SDC has been conducting research in the area of interactive graphic input/output for several years. Under ARPA sponsorship, an interactive computational graphics facility, The Assistant Mathematician (TAM), was developed. This experimental facility was designed to accept as input hand-printed mathematical expressions, which were then evaluated. (A hand-printed-character recognizer is an important and highly advanced component of TAM.) TAM has been viewed as a possible prototype for an on-line graphic programming system, a flowcharting system, and an automatic mathematics-typesetting system. It has also provided some basis for automatic signature verification and other potential intelligence applications. Throughout this work, two fundamental technologies have been steadily advanced: that of automatic pattern recognition and that of efficient and powerful man-machine interaction.

During the past year, at the direction of ARPA, these technologies were focused on interactive techniques for image and signal processing and enhancement--an area of acute importance in which technology has yet to provide means of processing that meet users' needs for speed, accuracy, and cost-effectiveness. Large amounts of image and signal data collected in the course of scientific experiments, medical tests, surveillance operations, and satellite and telescope photography must be processed by computer before they can be used effectively by human interpreters. Complete processing of high-resolution images, however, can require processing times measured in hours or days on even the fastest of modern computers. Interactive man-machine systems show considerable promise of reducing the amount of computation required to process image data by allowing human interpreters to determine dynamically the points in the processing of an image at which they, or the computer, should assume control of the processing strategy. In addition, the human can select processing strategies according to the known characteristics of the objects that are being searched for in an image, and can terminate a strategy when it becomes clear that the computer is pursuing a futile path. The Image and Signal Processing Project has explored the application of interactive man-computer communication techniques as a means of substantially reducing the time required for adequate processing of image data.

4.1 PROGRESS

During the year, the project's work focused on the following tasks:

1) Study of the discipline of image processing by computer.
2) Definition of a language and design of a computer system for interactive image processing.
3) Implementation of a model image processing Data Manager.
4) Design and implementation of a standard graphics access method to allow hardware flexibility in system implementation.

5) Conduct of an ARPANET graphics experiment with Massachusetts Institute of Technology.

These tasks are summarized below.

4.1.1 Study of Image Processing

Extensive reading and personal contact with individuals working in the various fields of image processing was necessary to provide background and a working environment for the project. A synopsis of the information gained was documented in [1]. The study covered image processing tasks (image enhancement, image analysis); image processing hardware and software functions; currently operating image processing systems analyzed according to operating system, system architecture, command languages, and interactive facilities; and the several programming languages now in use in image processing. The systems surveyed included VICAR (Video Image Communication and Retrieval), developed jointly by IBM and Jet Propulsion Laboratory; the ITEK system; the VL system located at the Visibility Laboratory at Scripps Institute of Oceanography; and a biomedical interactive system operated by the National Institutes of Health. The languages included FORTRAN, PAXII, Conversational PAX, and LIP1. The study report included initial evaluations of the capabilities of the system surveyed to carry out image processing tasks and functions in speedy and cost-effective ways. Although each system served a particular purpose acceptably, none was found well suited to the broad range of tasks that will be necessary in a truly interactive image-processing environment.

4.1.2 Interactive Image Processing System

After completing the survey of existing systems and languages, the project undertook the design of a system and a language that would overcome some of the limitations observed in systems currently in use. The primary goal was to provide an environment within which an image processing researcher can design processes, investigate results, and guide experimental steps. The SDC design provides a system that is interactive and natural: the user communicates in mathematical or graphic handwritten notation; he can create images or filters; he can isolate areas of interest; he can use standard tools or create his own. While all image processing systems are designed to manipulate the data contained in images, none has extensive capabilities for obtaining, storing, and retrieving information about images. The SDC design provides such a unique capability: it incorporates a catalog of attributes such as sensor system characteristics, parameters pertinent to the storage and handling of the image as a data set, and function descriptors. The significant features of the system design are listed below (separate reports [2], [3] provide details and examples):

- Interaction
    - On-line specification and control of processing
    - Creation and modification of algorithms, as-you-go selective display
- Natural Communication
    - Two-dimensional handwritten input
    - Mathematical notation for image processing specialist
    - Macro instructions for general user
    - Graphic input:
        - array generation
        - function definition
        - area isolation
- Unconstrained Data Management
    - Cognizance of attributes: image, sensor, environment, function, system, etc.
    - Interrogatable attribute catalog
- Automation
    - Automatic storage allocation
    - Automatic organization of computation
    - Attribute controlled processing, processing inferred from image, function, etc.
    - Parameters remembered
    - Automatic choice of and conversion between appropriate data types

4.1.3 Image Processing Data Manager

The project undertook to validate in detail the feasibility of one aspect of the interactive image processing system. To do this, a model Data Manager system was implemented. This system communicates with the user in a language that combines English and algebra and that permits selective image processing qualified by information housed in a dynamic, personalized data base. As well as serving its initial purposes, the Data Manager allowed us to study the data structures necessary for efficient storage and retrieval of pertinent information about images and processing programs.

To the Data Manager, an image is an object; objects belong to one or more abstract classes of objects. Thus, in the hierarchy in Figure 4-1, all the objects, M1, M2, V1, V2,..., L3, belong to the class ARRAY; M1 also belongs to the sub-classes PICTURE and MARINER.

A specific object is described by its attributes. Attributes are delineated in attribute-value pairs. For example, ROW=1000 is an attribute-value pair describing how many rows of pixels there are in picture M1.

These hierarchical constructs are the basis for retrieving data from the data base. Group, object, and attribute relationships can be expressed in simple data-base creation statements. To define the class structure expressed in Figure 4-1:

```
ARRAY: PICTURE, FILTER;
PICTURE: MARINER, VIKING;
FILTER: HP,LP;
MARINER: M1,M2,M3;
VIKING: V1,V2,V3;
HP: H1,H2;
LP: L1,L2,L3;
```

Attribute value pairs are similarly defined:

```
M1:  ROW:=1000,COL:=1000,FILTER:=H2,CHAR3:=S2;
V1:  ROW:=2000,COL:=2000,FILTER:=L2;
```

Very complex data relationships can be expressed with these forms. They are a flexible list structure which allows members of a list to point to other lists. With the attribute-value pairs, FILTER and CHAR3, the values are other lists, which may in turn point to still other lists. Note that attributes may be physical descriptors (such as ROW) or processing indicators (such as FILTER).

A query to the Data Manager is stated in the form of a conditional function. The function consists of two parts: the condition portion and the answer portion. The condition portion is a boolean expression which, in use, produces a list of names of data base entries satisfying the condition of the expression. The answer portion (optional) structures the Data Manager's response. It defines a series of arithmetic expressions which are evaluated; a value for every member of the list is returned.

```
boolean-expression [=>arithmetic expression [,arithmetic expression] ]
```

The answer portion is signaled by the arrow (=>). The word LIST or PRINT may be used in place of the arrow to serve the same function.

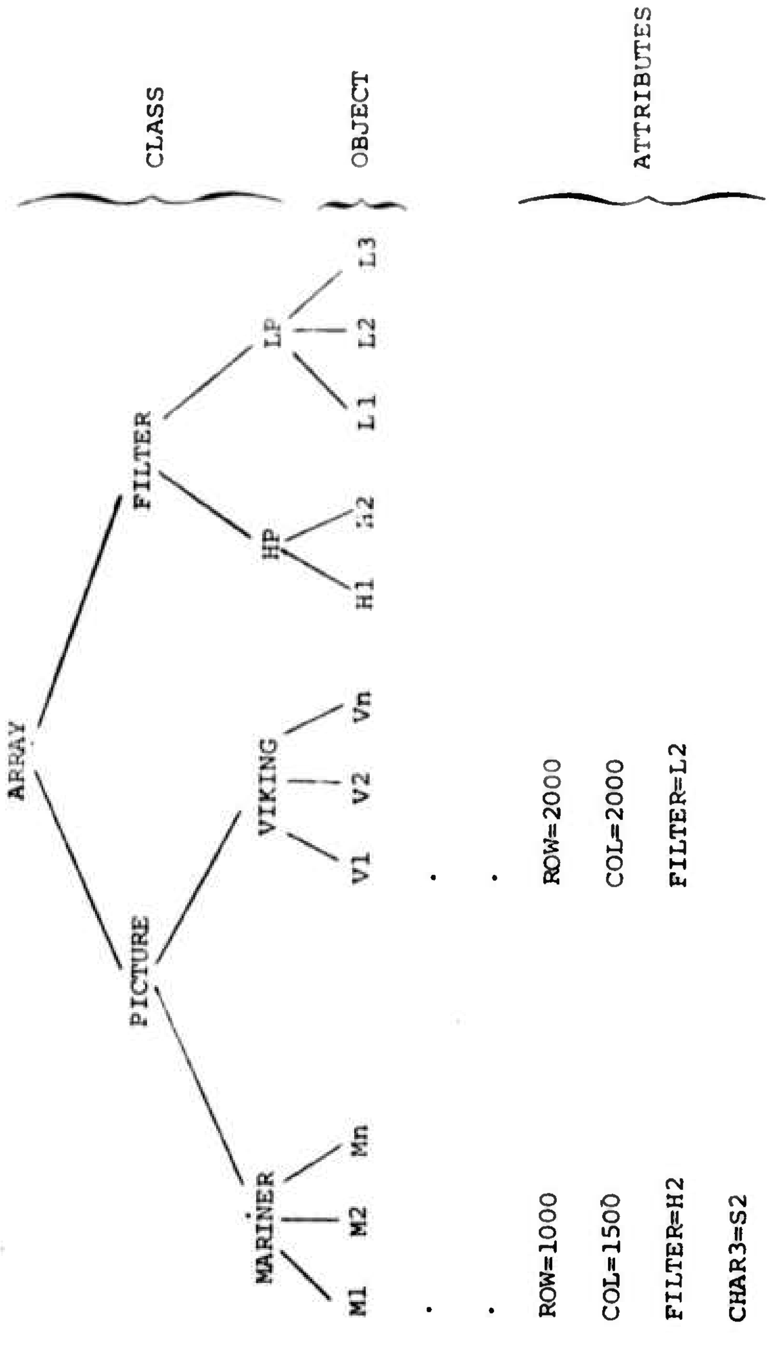


Figure 4-1. Hierarchical Structure of the Image Processing Data Base

If a query consists of only a boolean expression, the names of those objects satisfying the expression are returned (along with the last attribute of the expression).

For example: A query

```
PICTURE ROW > 2015
```

returns:

PICTURE	ROW
M2	2300
M5	3015
M6	2016

Since an arithmetic expression may be simply the name of an attribute, the answer portion can contain a list of attributes to be printed for each hit.

The Data Manager demonstrates the feasibility of a solution to the image processing data management problem. The hierarchical data base can be easily queried with an English-based language. The user can define his own arithmetic and boolean expressions (which may be saved in the data base). The job generation portion of the system demonstrates the feasibility of automatic retrieval of parameters from the data base and the direct execution of a complex task based upon a query statement with no further intervention by the user.

4.1.4 Standard Graphics Access Method

To allow maximum flexibility in selecting the hardware on which to implement the image processing system, to facilitate system implementation and component adaptation by localizing display equipment dependencies, and to facilitate network communication, a standard graphics-access method was defined and a package of subroutines, called the Graphic Support Module (GSM), was written to implement the method. GSM defines a standard graphic access method for a logical display and tablet. Programs using GSM can produce complex graphic output, including line drawings and character strings, and accept input in the forms of pen traces, single-point hits, and number line values from a tablet overlaid on the display screen. As presently implemented, GSM supports a Beta rear-projection, vector drawing display and a Science Accessories Corporation Graf Pen (sonic tablet). A Honeywell DDP-516 acts as an interface and controller between an IBM 370/145 and the graphic devices.

The GSM subroutines are a package of graphic utility routines that allow the writing of graphic applications programs with little or no knowledge of the physical display being used. An application program can be written in any language so long as parameter passing to the GSM follows FORTRAN linkage conventions.

GSM makes use of and depends on a free storage area being available when it is executed. A simulated IBM 360 Operating System GETMAIN routine provides a 32K byte free storage area for storing data lists and graphic buffers. FORTRAN programs also use this area for input and output buffers.

GSM is the communications link between the application program and the display control program operating in the Honeywell DDP-516. GSM initiates the display process by transferring the display control program from disk storage to the DDP-516.

Figure 4-2 shows all basic storage areas and communications links maintained by GSM.

4.1.5 ARPANET Graphics Experiment

During the early part of the contract year (September to November, 1972), an ARPANET graphics-communication experiment was conducted with the Dynamic Modeling and Computer Graphics Group at Massachusetts Institute of Technology. Hand-printed characters written at MIT were transmitted over the ARPANET to SDC, where they were analyzed with the SDC character recognizer; the results of the analysis were then returned to MIT via the ARPANET.

The purposes of the experiment were to use SDC's character recognizer to investigate the kinds of delays that might be expected when tablet data were transmitted across the network and to study methods by which character recognizers at various sites may be compared.

The experiment allowed MIT to transfer tablet data over the network in the format described in [5]. These data were converted to the form accepted by the SDC character recognizer, and "recognized" characters were returned over the network to MIT as character codes with size and position information. Each experiment session was conducted with a normal telephone voice link as well as the network computer link. Although it was not necessary to the experiment, all hand-drawn and recognized characters were displayed at both SDC and MIT to aid in debugging.

Fifteen experimental sessions were conducted. Problems encountered during these sessions included difficulties in opening and closing network connections, bad calls to host network utilities, and misunderstanding of data formats. After the initial problems were corrected, the experiment was run successfully: MIT tablet data were processed by the SDC character recognizer with an overall recognition rate of approximately 30%. A non-optimum character dictionary was used (one generated by the SDC personnel rather than the MIT personnel doing the writing), and this had much to do with the low recognition rate. The experiment was suspended when the SDC ARPANET graphics communication configuration ceased to operate satisfactorily.

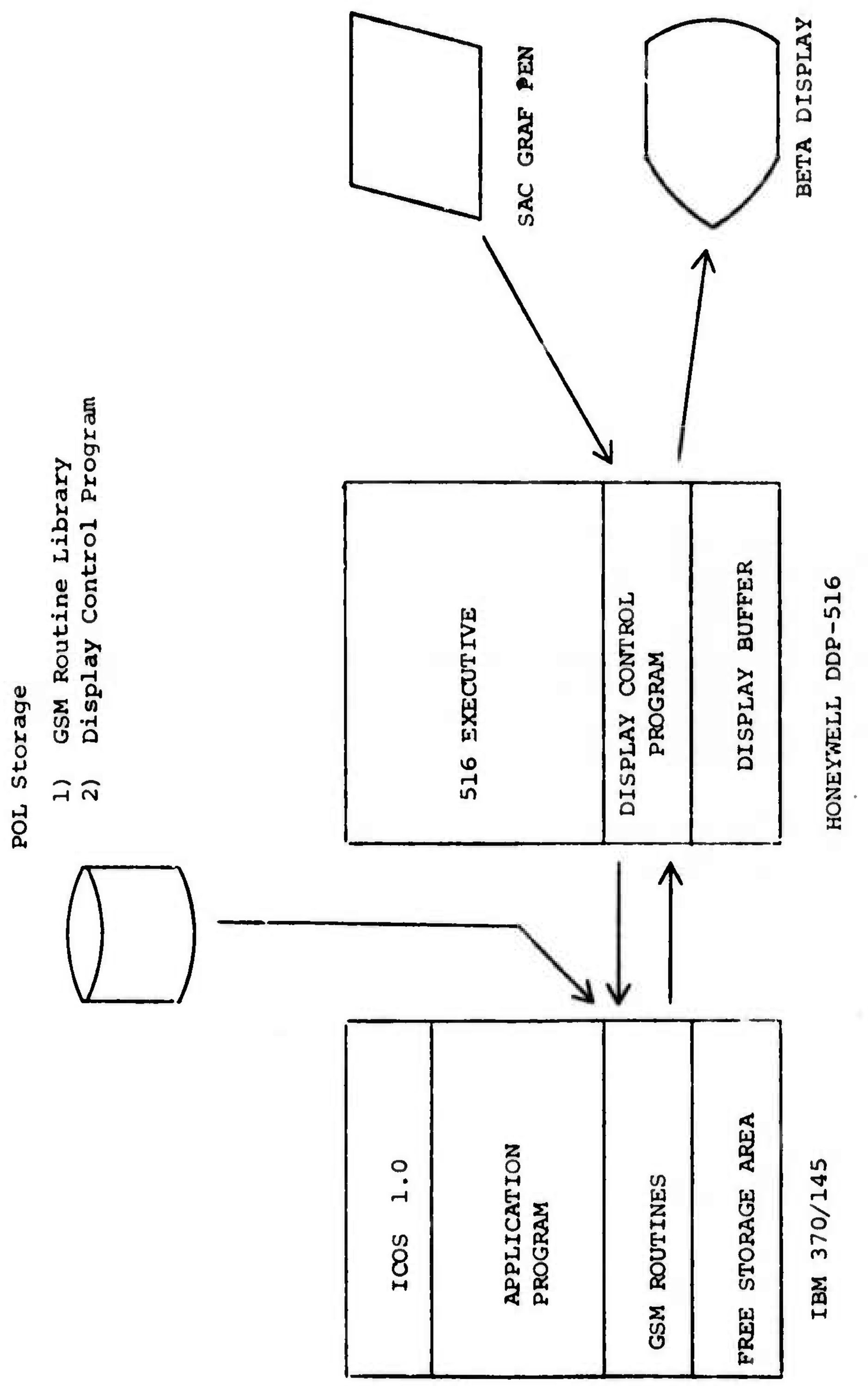


Figure 4-2. Graphic Support Module Configuration

The experiment showed that character recognizers can be tested and used over the ARPANET. Investigating delays over the network was difficult because of the general use of time-sharing systems at hosts. Therefore, no delays were researched. It was clear, however, that more work must be done if this process is to be practically useful: the problem of interaction between network and graphics facilities must be solved before any meaningful advances can be made at SDC with network graphics.

4.2 SUMMARY

The main emphasis of the past year's research in image and signal processing was on defining a data-processing environment suitable to the special activities and requirements of image analysts. The research included a survey and analysis of current image processing systems and their applications, a specification of the particular data processing requirements and interface characteristics of interactive on-line image processing, and the design of a new system and a new language optimally suited to these requirements and characteristics. The design is intended to allow an image analyst to direct the computer's processing of an image dynamically, specifying processing procedures and selecting alternatives on the basis of continuous information about processing progress. The design also includes an image processing "data manager" that adds to the basic system a capability for storing, manipulating, and retrieving environmental and contextual information about specific images.

In addition to activities specifically related to image processing, the project undertook a small-scale experiment, jointly with the Massachusetts Institute of Technology, in transmitting graphic-tablet data via the ARPA Network.

4.3 BIBLIOGRAPHY

[1] Bebb, Joan, Stromberg, W. D., and Nimensky, R. E., "An Overview of Image Processing," System Development Corporation Report TM-5068/001/00, Santa Monica, 1 March 1973.

[2] "An Image Processing System," System Development Corporation Report TM-5068/000/00, Santa Monica, 15 January 1973.

[3] Bebb, Joan, and Nimensky, R. E., "An Interactive Image Processing System: Design and Language Description," System Development Corporation Report TM-5068/002/00, Santa Monica, 1 April 1973

[4] Albrecht, D. E., "Graphic Support Module (GSM) User's Guide," System Development Corporation Report TM-5077/000/00, Santa Monica, 3 May 1973.

[5] Albrecht, D. E. "GSM Maintenance Manual," System Development Corporation Working Paper TM-5077/001/00, Santa Monica, 8 September 1973.

[6] Albrecht, D. E., "SDC-MIT ARPA Network Graphic Experiment," System Development Corporation Working Paper TM-5147/000/00, Santa Monica, 15 May 1973.

4.4 PROJECT PERSONNEL

J. E. Bebb, Project Leader
D. E. Albrecht
J. K. Igawa
R. E. Nimensky

5 NETWORK RESEARCH AND DEVELOPMENT

During the past year, major changes were made in SDC's computer laboratory configuration and operating system facilities. The Honeywell DDP-516 computer, which had served as an interface between the IMP and SDC's IBM 370/145 computer as well as a controller for 14 multiplexed terminal devices and a RAND tablet, proved to be inadequate for the increased load and complicated timing requirements being placed upon it. Consequently, preliminary work was begun to replace the DDP-516 with a PDP-11/05. At the same time, the Interactive Common Operating System (ICOS), which had served briefly as SDC's ARPA-program operating system after the 370/145 replaced the 360/67, was being replaced with IBM's VM-370. Although these changes slowed or interrupted the implementation of SDC's ARPANET facilities, work proceeded on finding the most feasible way of implementing a new Network Control Program on the modified facilities.

5.1 PROGRESS

VM-370 was modified to allow virtual machine communication by using pseudo-teletype terminals attached to a simulated IBM 2703 that is owned by the ARPANET virtual machine. These changes were made to four VM-370 programs to allow the Network Control Program to enable a user's virtual machine and to provide communication from this virtual machine to the ARPANET. A new local user TELNET facility was developed that provides local VM-370 CMS users with the ability to interface with remote hosts. This facility utilizes VM-370 spooling mechanisms to interface with the NCP machine.

The Network Control Program (NCP) at the University of California, Santa Barbara (UCSB), was determined to be the most feasible system to implement on the 370/145 under VM-370. The one deficiency of this system is that a full file-transfer capability is not implemented. However, the structure of the system and its high reliability made it very attractive. We received excellent cooperation from UCSB in procuring their system for our use.

The UCSB NCP has been slightly modified to be compatible with VM-370. Four new programs have been developed to interface with the system: (1) a logger routine that interfaces with VM-370 to provide virtual machine communication, (2) a file-transfer routine that provides file-transfer service, and (3) a user TELNET interface that uses the VM-370 spooling mechanisms to service local user TELNETs.

Routines were developed to interface the PDP-11/05 and the IMP and to service the 370/145. A Digital Equipment Corporation DX11B has been installed. The software package (COMTEX) for the system has also been installed. This package emulates an IBM 2848 controller, which can be used to service IBM 2260 display terminals or their equivalents. The interface between the PDP-11 software developed by SDC and the DEC-supported COMTEX package is in the development phase.

5.2 SUMMARY

The past year saw the evolution of plans and designs for a new SDC ARPANET interface system adapted from the system now operational at the University of California, Santa Barbara, and running on an IBM 370/145 computer under VM-370. The old DDP-516 programmable communications controller, which has served as the IMP-to-Host interface since April, 1970, is being replaced by a PDP-11/05 minicomputer.

During the coming year, the ARPANET Engineering project will focus its attention on refining the new network interface, developing a full set of network protocols encompassing file-transfer and mailbox facilities, and providing continuous support for the Speech Understanding Research project and the new Lexical Data Archive project.

5.3 PROJECT PERSONNEL

K. H. Brandon, Project Leader (ARPANET Engineering)
B. D. Warner, Project Leader (Systems Research)
R. Gates (part-time)
J. G. Hata
R. H. Larson
L. M. Molho (part-time)
D. L. Pintar (part-time)

6. NETWORK DATA MANAGEMENT

6.1 ENGLISH DATA ACCESS

Following a mid-year review by the Contract Technical Monitor, the CONVERSE project was restructured to address more directly the problem of English query-language capability for data management use. The resulting English Data Access (EDA) project was aimed at establishing a practical foundation for the design of a natural-language processor that could be used as an adjunct to existing data management systems, including particularly those that are based on hierarchical data structures. In emphasizing existing data management systems, this approach differs from that of the CONVERSE system, which incorporates its own data management subsystem.

The initial plan for the EDA project was to use the CONVERSE front-end natural-language compiler to process English statements. However, this plan presented two difficulties. First, the CONVERSE system's data management subsystem is based on relational data structures, and the concept structures used in the natural-language compiler reflect a relational organization. In attempting to interface the CONVERSE processor with existing hierarchical systems by representing hierarchical data bases in relational form, we encountered semantic problems that proved impossible to control; such representations change the meaning of data as viewed by a user, as well as the functions that he can perform on the data. Second, the CONVERSE system has been designed to be amenable to a wide variety of potential applications, and for that reason sacrifices processing efficiency for linguistic generality. In particular, CONVERSE relies heavily on syntax analysis of free-form English sentences. In an environment restricted to the vocabulary and limited syntactic complexity of data management, extensive syntactic analysis is unnecessary, and a semantically based technique, restricted to the data management lexicon, can be used instead, with considerable gains in processing efficiency. As a result, a semantically based technique was outlined in which the concepts embodied in data management system structures and functions are viewed as the objects of discourse. During the remainder of the year, this technique, given the name "Conceptual Processing," was elaborated and evaluated in light of similar approaches that have been taken by other investigators, notably Schank [1].

Most English language processors incorporate procedures that can be called "conceptual processing." The concepts that are modeled in such systems, however, are almost universally the concepts that are embodied in the structure of English: syntactic concepts. The technique proposed for EDA differs from other natural-language processing techniques in that the concepts reflect the structure, not of English, but of one or more data management systems to which a "Conceptual Processor" would be a dedicated adjunct. Structural and functional concepts reflect the data management environment, and semantic concepts reflect the subject content of particular data bases. As a result, there should be no need for the complex algorithms of conventional syntactic and semantic parsers. A natural-language processor based on this constrained discourse space should be efficient, practical to implement, simple to use, powerful in terms of comprehensive responses, and flexible in its use with existing data management systems.

6.1.1 Progress

The Conceptual Processing technique was defined and compared with other similar techniques. A detailed description of the technique, and the comparison with other English processors, is given in [2]. Its main elements are summarized below.

The Conceptual Processing technique involves, first, the use of structural and functional concepts. Structural concepts represent logical entities of hierarchical data structures and relations among them, as reflected in existing data management systems. Examples are data elements, repeating groups, items, columns, and rows. Functional concepts represent permissible functions that operate on the data structures (such as selection, output, and update). Work in this area was based chiefly on earlier work by the Network Data Sharing project and made direct use of the common language, called ILDS (Intermediate Language for Data Sharing [3]), as the target language into which English expressions are translated.

A Conceptual Processor must include semantic description mechanisms. These mechanisms are structures that reflect the human view of the data base. They describe the concepts a user is likely to use when directing English queries to the data base. They also describe the relationship between the semantic structure of a data base and its logical representation in the data management system, and they provide the basis for mapping English expressions into their equivalent ILDS representations. Other semantic mechanisms include properties of data base elements (human, numeric, etc.) and semantic referrals (word synonyms, idiomatic synonyms, etc.). Diller [4] presents a detailed analysis of the use of data base information in the Conceptual Processor. Other linguistic mechanisms include local use of English syntax (such as possessives and logical connectives), anaphora, and morphology. These mechanisms are used only if the semantic mechanisms do not suffice.

As presently envisioned, the Conceptual Processor would include the following major subsystems:

1) A conceptual network generator. This would be an interactive program that a systems analyst or programmer familiar with the target data management system would use to initialize a conceptual network for that system.

2) An input scanner. This program would read a user's input, sentence by sentence, and lay out identified sentential elements in a list or table format.

3) A dictionary and a dictionary-lookup routine. Words and word segments that are defined in the dictionary are replaced (in the list or table established by the input scanner) by their definitions. For most content words (i.e., words meaningful in terms of the conceptual network), a definition is a pointer to a concept (node) in the network. Function words ("of", "and", "what", "'s", etc.) have definitions that point to certain system subfunctions. Words that were not found in the dictionary (including numbers) would not be replaced.

4) A parsing algorithm. A variety of parsing procedures are available. The basic process, however, is to assemble the meanings of the words into a meaning for the sentence based on the relations between the concepts pointed to and the "definitions" of the function words. The output of this algorithm would be passed to the next stage: the translator.

5) Translator. The conceptual network, in addition to linking the meanings of the various concepts that can be talked about or referred to by the user, contains pointers to the functional and logical structures of the data management system from which it was derived by the conceptual network generator. Using this information, together with the syntax and semantics of the query language of the target system, the translator will output one or more well-formed queries to the target system.

6.1.2 Summary

For more than a decade, computer-science researchers at SDC and elsewhere have been attempting to construct practical and efficient programs for understanding and processing natural language, so that users who are not computing professionals could avail themselves of the computer's data management and information processing capabilities. To date, no true natural-language data management system has proved competitive with less flexible but more reliable and vastly more efficient formal-language systems.

SDC's CONVERSE system, which was developed in recent years under ARPA sponsorship, was designed as a research vehicle to explore the feasibility and cost of natural-language data management. By the beginning of this contract year, it had become apparent that the achievement of a general-purpose English data management system was still some distance into the future and that many practical and theoretical problems remained insolvable. In particular, much research remains to be done in applied linguistics and deductive inference.

At mid-year, at the direction of the ARPA Contract Technical Monitor, the CONVERSE project (also known as the English Input/Output project) was reorganized into two separate projects: the Deductive Inference Research project and the English Data Access project. The work on inference is discussed elsewhere in this report (Section 3). The English Data Access project was formed to capitalize on the experience of the CONVERSE project to date in an effort to package, or "engineer," a cost-effective solution to the English data management problem using present technology. During the remainder of the year, the project formulated the principles for a Conceptual Processor whose "conceptual world" is limited to the concepts--structures, functions, relationships--that are exhibited by or embodied in existing data management systems. The processor, which would be a logical and physical adjunct to a functioning data management system, would accept and "understand" only those English queries and commands that make sense in terms of the permissible functions of the data management system and in terms of the content of the data base being queried. By thus limiting the range of possible meanings a sentence may have, while still permitting it to be an unformalized English sentence, we hope to provide enough natural-language processing capability to make data management systems accessible to non-professional users without at the same time trying to solve the longer-range research problems involved in computer understanding of natural language.

6.1.3 Bibliography

[1] Schank, R., and Tesler, L., "A Conceptual Processor for Natural Language," Stanford AI Project Memo AI-76, Palo Alto, December, 1969.

[2] Shoshani, A., and Burger, J., "Conceptual Processing of English for Data Management Systems," System Development Corporation Working Paper TM-5207, Santa Monica, 17 August 1973.

[3] Shoshani, A., and Spiegler, I., "The Integration of Data Management Systems on a Computer Network," AIAA Computer Network Systems Conference, Huntsville, Alabama, April 1973. (Also NIC #15717.)

[4] Diller, T., "Data Base Information on English Words in Q-A System Queries," System Development Corporation Report TM-5227, Santa Monica, 15 October 1973.

6.1.4 Project Personnel

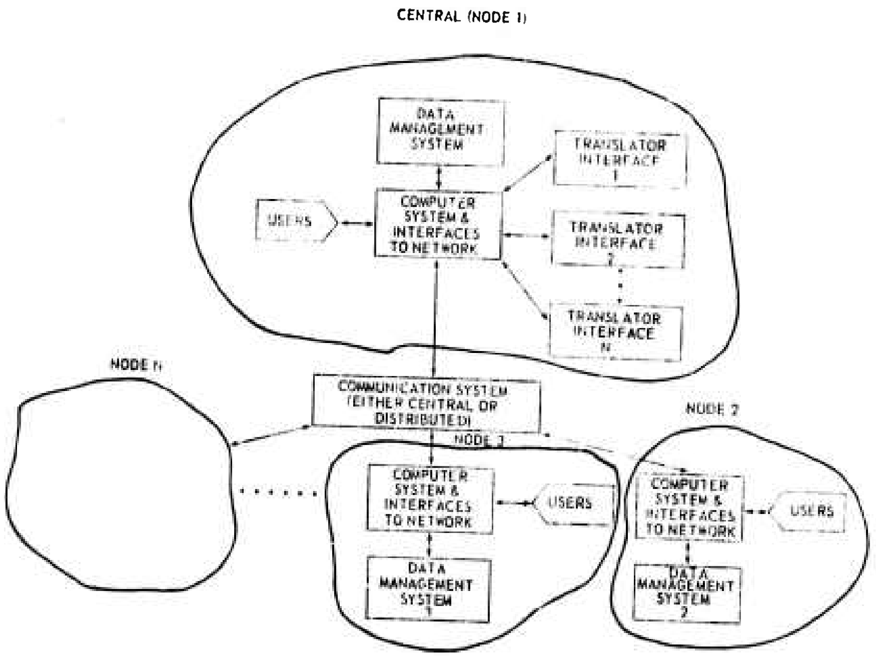
A. Shoshani, Project Leader
J. Burger
T. Diller
A. Leal
I. Spiegler

6.2 NETWORK DATA SHARING

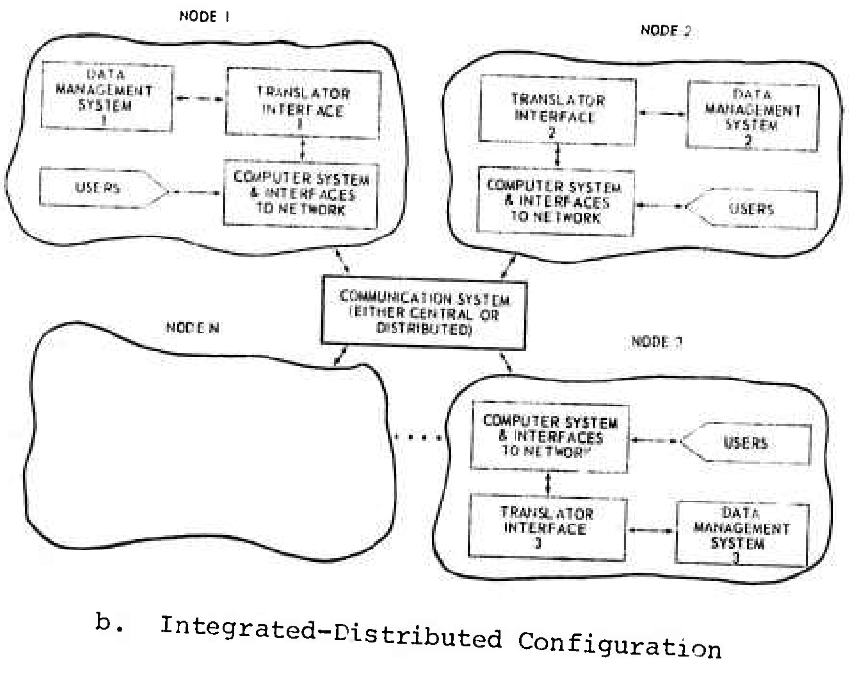
The purpose of the Network Data Sharing project, which concluded this year, was to investigate software techniques that would give any user on a computer network the capability of accessing data located in computers other than his own throughout the network. In our view, a major advantage of such a capability would be the elimination of the need for duplication of data bases within different host computers around the network or for manual transshipment of data in the form of card decks, tapes, or printouts. The chief savings would be in the time required to access important data at a remote location. For purposes of this project, it was assumed that the types of computer networks at issue are those that might be used by agencies in which data security within the network is outweighed by the efficiency and economy of shared access--for example, agencies such as logistics commands or the Veterans Administration. (We have, however, considered possible implementation of secure data-sharing facilities, as is noted below.) It was also assumed that a feasible approach to data sharing must be one that (1) does not require the rebuilding of disparate data bases or data management systems to a single design and (2) does not result in interruption of the continued use or updating of existing data management facilities.

Using these assumptions as guidelines, the project chose, from among several possible technical approaches to data sharing, one called the integrated approach [1]. The integrated approach posits a single network data sharing language--called the "common" language--and translator interfaces, accompanying each data management system in the network that would perform translations between the common language and the local data management languages. Queries would be expressed in the common language, transmitted to the target host's translator interface, translated into the host's local query language, and applied to the appropriate local data base(s). The requested data would then be passed back to the requester either directly, by the target data management system, or through the translator interface and via the common language. (The latter option would be useful when a request resulted in the accessing of data distributed among two or more data bases, possibly at two or more hosts on the network.)

The integrated approach to data sharing is susceptible to two network-configuration variations: the integrated-distributed and the integrated-central; these are diagrammed in Figures 6-1a and 6-1b. In an integrated distributed configuration, the interface for each host's data management system would be physically located at the host site; in an integrated-central configuration, all of the translator interfaces for all data management systems on the network would reside in a central dedicated host node. There are advantages to both configurations. An integrated-distributed network, for example, would be fail-soft, since the destruction of a single node could not cause a complete disruption of communication within the network. The integrated-central configuration, on the other hand, could be less expensive to implement and could be adapted to provide centralized security.



a. Integrated-Central Configuration



b. Integrated-Distributed Configuration

Figure 6-1. Integrated Data Sharing Network Configurations

During the previous contract year, the project's efforts were concentrated on developing a common language and building experimental translators from the common language to example data management systems. The development of the common language involved a research effort in determining the most appropriate data structures on which to base the language and defining the language's functional properties. This led to the definition of the language's syntax, which was used for the translators. Experiments in building translators were carried out to determine the feasibility of the technical approach and to uncover problems of implementation. An SDC metacompiler system was used to implement the experimental translators.

The common language developed during the previous year was based on relational data structures, which were chosen because of their generality and relatively simple structure. Experiments with translators for these structures revealed two serious difficulties in attempts to translate from the relational common language to data management languages based on hierarchical data structures (such as TDMS). First, some functions that were easily expressible in the hierarchical data management language required highly complex expressions in the relational common language, making the translator itself needlessly complicated and its use impractical. Second, there is a semantic difficulty in trying to represent the logical structure of a hierarchical data base in terms of an equivalent relational data base. The representation changes the meaning of the content of data as viewed by a user, as well as the functions he can perform on the data.

As a result of these experiments, the design goal for this year was to develop a common language based on hierarchical data structures and to implement an experimental translator from it to an example hierarchical data management language. The language was termed ILDS (Intermediate Language for Data Sharing).

6.2.1 Progress

The year's work involved two main activities: (1) studying data management functions in order to isolate functional properties that would be most desirable in the common language, and (2) making further studies of the translator interfaces so that practical experience would be available to indicate their feasibility and cost.

The study of data management functional properties for ILDS relied heavily on the surveys published by MITRE [2] and the CODASYL Systems Committee [3]. The study [4] led to the specification of a kernel version of the language [5] with four functions: create, delete, update, and query. These functions are described in terms of three basic elements: qualifier, output, and replacement. The qualifier consists of functions that can be applied recursively to a hierarchy to determine what <u>entries</u> qualify. The entries that qualify can be operated upon by an output function to produce values, or by a replacement function to change values in those entries. Thus, the create function has only a replacement part, which specifies how to create new entries; the delete function has only a qualifier part, which selects entries to delete; the update function has a qualifier part which selects entries to be modified according to a replacement part; and the query function has a qualifier part which selects entries on which an output part operates to return values.

In the kernel version, the qualifier part is most powerful, since it is composed of recursive applications of elementary "entry-functions" (efunction in the syntax of ILDS). The entry-functions are select, group, scope, compare, loop, and logical qualification. The significance of the entry-functions is that they represent the most elementary functions for qualifying entries in a hierarchy.

The kernel version of ILDS was not an attempt to represent all possible functions on a hierarchical data base; rather, it provided us with a powerful enough tool for experimentation with translation into target data management languages. In the course of developing ILDS, we recognized many functional properties that were not expressed in the kernel version.

Several translator interfaces were implemented in order to gain experience, possibly identify unforeseen problems, and develop a better idea of the feasibility of the integrated approach to data sharing. One translator is described in [6]. Some of the functions that an interface must perform are described below.

- Translation from the common language (ILDS) to the target data management language (DML). The translation process has to be tailored for the specific DML, in that it needs to take advantage of special functions that can be expressed by the DML. If there are two ways of representing a request in the DML, the more efficient one must be chosen. In the case of ILDS, we have to consider whether a qualifier with multiple entry-functions can be expressed jointly through a single request in the DML.

- Appropriate refusal of a request. If the target data management system is not capable of performing a request expressed in the common language, then the interface needs to refuse it properly. Rather than refusing it as "cannot be done," an indication as to why it cannot be done might help the user to rephrase his request.

- Translation of returned data. Replies from the data management system must be put in a common data format if there is a need to integrate replies coming from different systems (this point was mentioned earlier). Furthermore, a common data format is necessary for returning complex data such as reports. A report that is produced by a system in order to be output on some local I/O device needs to be shipped to another remote system that might have different I/O devices. A "post-processor" at the remote node would then receive the report in the common data format and output it on an I/O device designated by the user who issued the request. If general mechanisms for data transfer are available on a computer network, then the interface should use them for returned data.

- Translation of error messages into a common error format. Every data management system usually has its own error-message format. It is useful to have them translated into a common format so that users have to be familiar with only one error format.

- Control functions. The interfaces need to have control mechanisms to handle multiple requests coming from different users on the network. This type of control mechanism should normally be part of network functions performed by every node. An additional control mechanism is necessary in order to establish communication between the interface and the target data management system.

Our experiments with implementing interfaces concentrated on translation from the common language to the target data management languages, because we wanted to learn more about this process of translation and its implications. An important conclusion was that if temporary storage functions (or their equivalent) are available in the target data management system, then it is possible to enhance the apparent functional power of that system by splitting complex requests in the common language into a series of requests in the DML. For example, in the case of DS/2 (an SDC data management system) a "SUBSET qualifier" function restricts the data base for the next operation according to the qualifier. The translator to DS/2 can take advantage of this function by representing, say, a complex query request as a series of SUBSET functions followed by a simple request that can be expressed in the DS/2 DML.

6.2.2 Summary

During the past two years, this project has conducted research whose results suggest that the integration of existing data management systems on a computer network might be achieved by using the technique of a common language and translation interfaces. This technique offers many advantages. It should lead to sharing of existing and new data, to gradual acceptance of the computer network data sharing concept, to the natural survival of better data management systems and elimination of less useful, less efficient ones, and to the elimination of duplicated work in the area of data management system development. Its feasibility is predicated on a well-defined computer network with advanced interprocess communication and data transfer capabilities.

A potential difficulty of this approach is that translator interfaces add a new level of language translation to the data management process. However, our experience showed that the translation of data management languages is a manageable task. Furthermore, since control, communication, and data transfer functions need to be performed in any approach to data sharing on a network, the additional task of translation may not be too significant.

The common language should represent a "union" of data management system query languages, so that it can represent requests for all data management systems. Our approach was to identify the elementary functions on logical data structures and allow for their recursive application. A kernel version of this language (ILDS) was defined.

Experimental translators from ILDS to example data management languages were built by the use of a metacompiler. These translators were relatively easy to implement (two to three man-months per translator).

Although the project was concluded this year, much of the conceptual work is continuing as part of the Common Information Structures project, which is described in Section 6.3.

6.2.3 Bibliography

[1] Shoshani, A., and Spiegler, I., "The Integration of Data Management Systems on a Computer Network," AIAA Computer Network Systems Conference, Huntsville, Alabama, April, 1973.

[2] Fry, J. P., et al., "Data Management Systems Survey," MITRE Corporation MTP-329, January, 1969.

[3] "Feature Analysis of Generalized Data Base Management Systems," CODASYL Systems Committee, April, 1971.

[4] Shoshani, A., and Spiegler, I., "Data Management Features for Data Sharing: A Study Based on MITRE's Survey," System Development Corporation Working Paper TM-5230, Santa Monica, in press.

[5] Shoshani, A., Spiegler, I., and Luther, G., "A Common Language for Data Sharing Based on Hierarchical Data Structures," System Development Corporation Working Paper TM-5231, Santa Monica, in press.

[6] Shoshani, A., and Spiegler, I., "The Translator from a Hierarchical Intermediate Language (HILDS) to TDMS," System Development Corporation Working Paper TM-5232, Santa Monica, in press.

6.2.4 Project Personnel

A. Shoshani, Project Leader
I. Spiegler

6.3 COMMON INFORMATION STRUCTURES

The need for the transfer of data bases among information systems grows constantly as more and more data are collected and manipulated by computers. The introduction of new, more advanced information management systems requires that data bases created and manipulated on old systems be reshaped and transplanted into equivalent data bases for new systems. (By "information management systems," we mean any computing environment supporting data base retrieval and generation.) This process is usually inefficient, slow, and expensive, especially for large data bases. The need for data base transfer also arises in situations where the data that exist in one application system could be useful for a second application system, provided that they could be restructured to be operated on by the second system.

The Common Information Structures project is attempting to develop a methodology and techniques for restructuring and transferring data bases across disparate information systems. Information system data structures, in their physical organizations and embodiments, reflect the uniqueness of the system applications. Each system employs data structure organizations, data access methods, and data management functions that are specifically tailored to a particular application. As a result, one information system cannot readily access data contained in another information system, particularly when the two systems function within different operating environments. The project's goal is to make wider access possible by (1) describing data bases in terms of data structure levels and (2) transferring data bases from one system to another through these levels. The approach being taken is to use logical data structure characteristics as a basis for a global transfer technique that will allow information systems to use data bases that were previously incompatible.

6.3.1 Progress

Initially, this research was directed toward an analysis of information system data structures in order to gain some insight into data structure characteristics. An analysis of the literature ([1]-[13]) led to the separation of data structure characteristics into three data structure levels: logical, storage, and physical. The users of an information system should be given the capability to describe data bases at a logical level, independently of both the system's internal design and the characteristics of the computer hardware. Data structure characteristics that reflect logical entities and the relationships among these entities, as viewed by a system user, constitute the logical data structure level, as well as reflecting the view that the end user of such data has of it, being completely uncoupled from its realization as a component within an information system. The storage data structure level comprises data characteristics that effect data inversions and indexing organization of the data. System designers manipulate storage data structure characteristics to achieve space-time efficiency. These characteristics may be peculiar to a

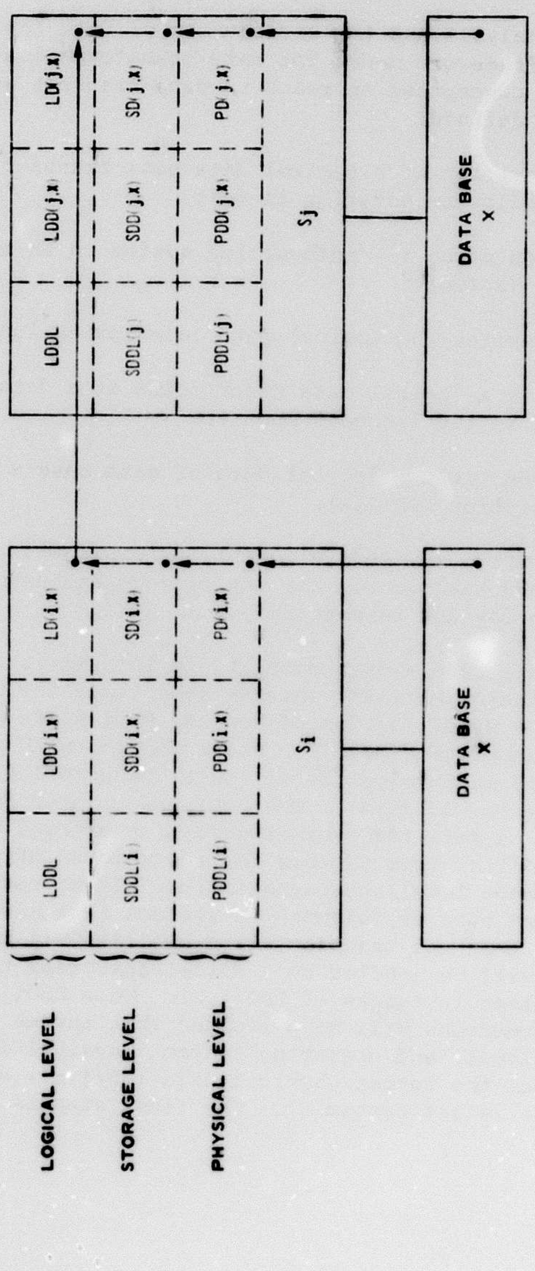


Figure 6-2. Data Base Transfer Through Data Structure Levels

specific information system. The physical data structure level comprises data characteristics that are intimately connected to the operating system. Details of this analysis are contained in a report by Fogt [14].

On the basis of this analysis and the structuring of data representations into three levels, a basic framework model for data transformation through these data structure levels and a conceptual approach to data transfer across information system boundaries were defined.

Figure 6-2 is an illustration of a typical data base transfer. For convenience and conciseness, the following notation is used:

- S_i represents the i^{th} information system in an environment of n systems $(2 \leq i \leq n)$.
- LDDL(i) denotes the logical data description language of S_i.
- LDD(i,x) is a logical data description of a data base x in S_i expressed in the language LDDL(i).
- LD(i,x) denotes the logical data of data base x according to the description LDD(i,x).
- SDDL(i), SDD(i,x), SD(i,x), PDDL(i), PDD(i,x), and PD(i,x) are defined analogously for the storage and physical data structure levels, respectively.

To extract physical data from a data base, x, in information system S_i, we read data base x, producing PD(i,x). At the same time, the physical data description PDD(i,x), a specification of how the PD(i,x) is organized, is also created. A transformation must now be performed on PD(i,x) and PDD(i,x), removing operating system (physical level) data characteristics, in order to extract SD(i,x) and create SDD(i,x). The SDD(i,x) created describes how the SD(i,x) is organized. A data transformer must then extract LD(i,x) from SD(i,x) and create LDD(i,x) from SDD(i,x) by the removal of information-system-dependent (storage level) characteristics. If we restrict our attention to (or create) the rare case of information systems that have common logical data characteristics, then the transformation LD(i,x)→LD(j,x) is trivial. The usual case, however, will be handled by a data transformer that maps instances of LDD(i,x) to equivalent instances of LDD(j,x). Once LDD(j,x) and LD(j,x) are obtained, transformations will be performed that invoke necessary information system (storage level) and operating system (physical level) data characteristics so that the target physical data (PD(j,x)) will be an efficient representation for the target system S_j. The final step is to write PD(j,x) as data base x on S_j.

In summary, the data structure levels are used to modularize the data transformation process. Three data description languages are defined: logical, storage, and physical. Each is a formalism to describe the relationships between its corresponding data structures and data. Data are transformed from physical to storage, storage to logical, logical to storage, and storage to physical data structure levels by data transformers.

To implement this data transfer model, a methodology for the following tasks is being developed:

1) Remove machine and system dependent data characteristics from a source file.

2) Transfer source logical data to target logical data.

3) Insert system and machine dependent data characteristics into the target file.

4) Automate the above phases.

The project is currently directing attention to the logical-level transfer techniques, since they are so vital to the transfer process. The approach that has been developed is outlined below. First, some additional notation is introduced to precisely express the tasks involved:

- FDL(i) represents the file description mechanism (as it exists) in S_i.
- LDDL(i) denotes the logical subset of FDL(i) that is relevant to the data transfer process.
- The set of all possible data base descriptions LDD(i,x) in system S_i is denoted LDD(i).
- A data description mapping, which specifies for every instance of LDD(i) an <u>equivalent</u> instance of LDD(j), is called M_{ij}.
- A data transformer, LD(i,x)$\rightarrow$LD(j,x), is denoted $T_{ij}(x)$.

The data transfer process can be envisioned as beginning with the derivation of an LDD(i,x) in the language LDDL(i). The data description mapping M_{ij} will then specify an equivalent LDD(j,x), given this LDD(i,x). The logical data LD(i,x) will be extracted from the data base x using the query capabilities of S_i and the instances of LD(j,x), which will then be written in S_j using the S_j file generate capabilities and the LDD(j,x).

Two alternative approaches to specifying data description mapping are being considered:

1) Let S_i and S_j represent the i^{th} and j^{th} systems ($i \neq j$) in an environment of n information systems. There are $n(n-1)$ mappings (M_{ij}) necessary in order to map every LDD(i) to every LDD(j). Two mappings are necessary for every pair i,j:

$$\begin{tikzcd} \mathrm{LDD}(i) \arrow[r] & \overset{M_{ij}}{\boxed{\cdot}} \arrow[r] & \mathrm{LDD}(j) \end{tikzcd}$$

$$\begin{tikzcd} \mathrm{LDD}(j) \arrow[r] & \overset{M_{ji}}{\boxed{\cdot}} \arrow[r] & \mathrm{LDD}(i) \end{tikzcd}$$

In most instances, M_{ij} is not the same as M_{ji}, since data structure types and constraints will differ between the systems. However, we should be able to take advantage of any commonalities found (from the studies in previous steps) in the generation of the M_{ij}'s.

2) An alternative (and probably more efficient) way to achieve mappings between the data descriptions is by identifying the commonality intrinsic to logical data structures and logical data descriptions and defining a common logical data description (CLDD) and its corresponding language (CLDDL). (The possibility of making the CLDDL some set-theoretic formulation of all the DDLs is being explored.)

Let M_{ic} be a data description mapping from LDD(i) to a CLDD. Two typical mappings are necessary for systems S_i and S_j:

$$\begin{tikzcd}
\text{LDD(i)} \arrow[r] & \boxed{M_{ic}} \arrow[r, "\text{CLDD}"] & \boxed{M_{cj}} \arrow[r] & \text{LDD(j)} \\
\text{LDD(j)} \arrow[r] & \boxed{M_{jc}} \arrow[r, "\text{CLDD}"] & \boxed{M_{ci}} \arrow[r] & \text{LDD(i)}
\end{tikzcd}$$

In this approach, only 2n mappings are necessary. However, this method requires specification of a CLDDL, whereas the first approach requires no such preliminary step. The more commonality we find in the logical data structures, the more efficient the second approach is.

These results are presented in detail in the report cited earlier [14].

We are currently studying SDC's DS/3 system and IBM's IMS system in order to implement prototype logical level transfer techniques for them. We are also analyzing the works of the CODASYL SDDTTG and its members in order to ensure that our approach incorporates problem solutions that they have identified.

6.3.2 Summary

During this year, the Common Information Structures project defined a data transfer model that will allow information system users to access, in familiar environments, previously inaccessible files. The model is based on techniques for structural transformation of data through three data structure levels: logical, storage, and physical. The major research focus now is to characterize the essential elements of logical data description language mappings and logical data transformers.

Work during the coming year will be directed toward implementing these mappings and transformers for some sample systems. The following tasks appear to be necessary in order to develop practical techniques for data base conversion:

1) Develop a logical data description language that captures the semantic properties of the information in data bases. Semantic properties reflect the user's expectation as to the performance efficiency of the data management system and the functions he is most likely to request of it.

2) Define the process by which the logical descriptions of the source and target data bases are arrived at. At first, this task will be done manually by using a description of relevant characteristics of the source and target data management systems.

3) Generate a data translator, given the source and target logical descriptions as well as the correspondence between them. This is a major task that will assume the existence of an intermediate data format. The intermediate format will be developed by using known concepts (if possible), such as a normalized relational data structure.

4) Extract the source data base into its corresponding intermediate form using the source system query capabilities, and generate the target data base from the translated intermediate form using the target system generate capabilities. At first, this task will be performed manually for some example data bases and data management systems in order to assess its complexity.

6.3.3 Bibliography

[1] Fry, J. P., Smith, D. P., and Taylor, R. W., "An Approach to Stored Data Definition and Translation," Proceedings 1972 ACM SIGFIDET Workshop, December, 1972, pp. 13-57.

[2] Fry, J. P., Frank, L., and Hershey, E. A.,III, "A Developmental Model for Data Translation," Proceedings 1972 ACM SIGFIDET Workshop, December, 1972, pp. 77-107.

[3] Smith, D. P., "A Method for Data Translation Using the Stored Data Definition and Translation Task Group Languages," Proceedings 1972 ACM SIGFIDET Workshop, December, 1972, pp. 107-125.

[4] Astrahan, M. M., Altman, E. B., Fehder, P. L., and Senko, M. E., "Concepts of a Data Independent Accessing Model," Proceedings 1972 ACM SIGFIDET Workshop, December, 1972, pp. 349-363.

[5] Astrahan, M. M., Altman, E. B., Fehder, P. L., and Senko, M. E., "Specifications in a Data Independent Accessing Model," Proceedings 1972 ACM SIGFIDET Workshop, December, 1972, pp. 363-382.

[6] Taylor, R. W., "Generalized Data Base Management System Data Structures and Their Mapping to Physical Storage," Ph.D. Dissertation, University of Massachusetts, 1971.

[7] CODASYL Data Base Task Group Report, April, 1971.

[8] Senko, M. E., Altman, E. B., Astrahan, M. M., and Fehder, P. L., "Data Structures and Accessing in Data Base Systems," IBM Systems Journal, Vol. 12, No. 1 (1973), pp. 30-93.

[9] Language Structure Group of the CODASYL Development Committee: "An Information Algebra," Communications of the ACM, April, 1962, pp. 190-204.

[10] Engles, R. W., "A Tutorial on Data Base Organization," Annual Review of Automated Programming, Vol. 7, Part 1, 1972.

[11] Huang, J. C., "An Algebraic Model for Data Base Management Systems,"

[12] Fry, J. P., et al., "Data Management Systems Survey," MITRE Corporation MTP-329, January, 1969.

[13] "The Debate on Data Base Systems," EDP Analyzer, Vol. 10, No. 3, March, 1972.

[14] Fogt, K. J., "Data Structure Levels in Information Systems," System Development Corporation Report TM-5123, Santa Monica, June 28, 1973.

6.3.4 Project Personnel

Dr. A. Shoshani, Project Leader
K. J. Fogt